puppet by Perforce

Comply

Contents

Welcome to Puppet Comply

Puppet Comply is a tool that assesses the infrastructure you manage with Puppet Enterprise against CIS Benchmarks — the best practices for securely configuring systems from the Center for Internet Security (CIS).

Using Comply, you can:

- Run scans to check the compliance of your infrastructure against CIS Benchmarks.
- Set your desired compliance — a default benchmark and profile that you want your scans to be measured against.
- Customize profiles to specify which rules you want visible in scan reports.
- Identify the cause and source of compliance failures, and determine what configuration changes must be made to which systems.

Comply uses Puppet Enterprise (PE) to retrieve node and fact information. Once you have installed Comply, you must configure it to integrate with PE.

If this is your first time using Comply, try out our Beginner's guide to Comply.

Puppet Comply docs links	Other useful places
Learn the basics: Comply overview Comply terminology Beginner's guide to Comply Release notes **Install and configure Comply:** System requirements Install Puppet Application Manager Set up Comply **Run and manage CIS scans:** Run a CIS scan Set desired compliance Create a custom profile on page 95 View scan results	**Comply videos:** Comply introduction and demo **Docs for related Puppet products:** Puppet Enterprise Puppet Forge **Get help:** Troubleshooting Support portal

Comply overview

Welcome to the Puppet Comply!

This overview is intended for new users of Comply. We go over what Comply is, how it works, and show a demo of the 1.0.0 release. Before you begin, we recommend familiarizing yourself with our terminology.

What is Comply and how does it work?

Comply is a tool that expands the compliance capabilities of Puppet Enterprise (PE), by integrating with the *CIS assessor* to scan your infrastructure against the latest *CIS Benchmarks*. Comply connects to your PE environment and gathers information about your PE managed nodes, including operating system facts and classification node groups. It uses this information to suggest appropriate scans.

You can choose to run ad-hoc scans or *desired compliance* scans — a default CIS benchmark and profile scan that you assign to a node. Comply can automate desired compliance for you based on the information it gathers about your nodes from PE, or you can manually choose your desired compliance from a list of benchmarks and profiles. You can also create *custom profiles* to fit internally defined standards, by specifying which rules you want visible in scan reports. Most of the time, you only need to set your desired compliance once.

The scans are run as a *task* in PE. Scan results populate in the Comply **Compliance dashboard**, where you can see the number of nodes scanned and their compliance breakdown. In each node listed, there is a further breakdown of rule information which tells you why that rule is important, and steps you can take to fix the rule if it is failing the scans.

To see Comply in action, watch the demo below, or go through the steps yourself in our getting started guide.

For a full list of features, see the release notes.

Comply demo

The following demo walks you through the key features of the Comply 1.0.0 release:

Comply terminology

Key terms to be familiar with when using Puppet Comply.

CIS Benchmarks

Developed by the Center for Internet Security (CIS), *CIS Benchmarks* are internationally recognized standards and best practices for securely configuring systems. For more information, see CIS Benchmarks.

CIS assessor

Comply integrates with the *CIS assessor* (CIS-CAT PRO), the scanner tool that assesses CIS benchmarks. As part of the Comply configuration process, Puppet Enterprise (PE) installs the CIS assessor on your target nodes. For more information on the assessor, see CIS-CAT Pro.

Profiles

CIS Benchmarks include different levels of security settings, called *profiles*. The *Level 1* profiles are the base recommendation for every system, and the *Level 2* profiles are intended for environments requiring greater security. Comply can scan for either profile.

Rules

Each profile contains multiple *rules* that define specific elements of system configuration.

Custom profiles

A *custom profile* is a benchmark profile that you customize to fit your organization's internally defined standards, by specifying which rules you want visible in scan reports. Once you create a custom profile, it appears as an option in Comply when selecting a benchmark and profile.

Desired compliance

Desired compliance is the benchmark and profile that you assign to a node. It becomes the default scan for that node.

For a full list of Puppet terminology, see the Puppet Glossary.

Supported CIS benchmarks

Comply supports the following CIS operating system benchmarks.

Operating system	Supported versions
Amazon Linux	2
CentOS	6, 7, 8
Debian	8, 9, 10
Oracle Linux	6, 7, 8
Red Hat Enterprise Linux (RHEL)	6, 7, 7 STIG, 8
SUSE Linux Enterprise Server (SLES)	12, 15
Ubuntu	16.04, 18.04, 20.04
Windows	2012 R2, 2016, 2016 STIG, 2019, 2019 STIG, 10
Mac OS X	10.14, 10.15

Release notes

New features, enhancements, known issues, and resolved issues for the Comply 1.x release series.

- Comply release notes on page 6

These are the new features, enhancements, and resolved issues for the Puppet Comply 1.x release series.

- Comply known issues on page 9

These are the known issues for the Comply 1.x release series.

Comply release notes

These are the new features, enhancements, and resolved issues for the Puppet Comply 1.x release series.

Comply 1.0.4

Released May 2021.

New in this release:

- **CIS-CAT Pro Assessor v4.6.0.** Comply 1.0.4 includes the latest version of the CIS-CAT assessor and its associated benchmarks:
  - CentOS Linux 7 v3.1.0
  - Microsoft Windows Server 2019 Benchmark v1.2.0
  - Microsoft Windows Server 2019 STIG Benchmark v1.0.0
  - Red Hat Enterprise Linux 7 Benchmark v3.1.0
  - Red Hat Enterprise Linux 7 STIG Benchmark v1.0.1
  - SUSE Linux Enterprise Server 12 Benchmark v3.0.0
  - Ubuntu Linux 20.04 LTS Benchmark v1.1.0
- **Windows 2016 Datacenter.** The Windows 2016 Datacenter is now available as a desired compliance benchmark.
- **Updated module dependencies.** The `comply` module now includes the latest dependency releases.

Resolved in this release:

- **License check errors.** This release fixes an issue where the licence check returned an error if you installed Comply at the same time as Continuous Delivery for PE.

Security notice:

- **Vulnerability in `bluemonday` dependency.** This release updates the `bluemonday` package to version 1.0.9.

Comply 1.0.3

Released April 2021.

New in this release:

- **CIS-CAT Pro Assessor v4.4.0.** Comply 1.0.3 includes the latest version of the CIS-CAT assessor and its associated benchmarks:
  - CentOS Linux 6 v3.0.0
  - Microsoft Windows 10 Enterprise Release 20H2 v1.10.0
  - Oracle Linux 6 v2.0.0
  - Red Hat Enterprise Linux 6 v3.0.0
- **Mac OS X benchmark support**. Comply now supports Mac OS X 10.14 and 10.15 benchmarks.
- **Windows 10 Enterprise benchmark support.** Comply now supports Windows 10 Enterprise benchmarks. Note that these are compatible with Windows PRO.
- **The `oauth2-proxy` file server v7.1.1.** The `oauth2-proxy` image, that provides authentication in Comply, has been updated to version 7.1.1.
- **Benchmark name displayed in tables**. Comply now includes the benchmark name in the **Desired compliance set** column on the **Inventory** page.
- **Updated navigation icon for Inventory.** Comply has a new custom icon for **Inventory** in the side navigation bar.

Resolved in this release:

- **Node results table shows incorrect time.** This release fixes an issue in the node results table that showed the last scan as being an hour behind the current time.

Security notice:

- **Vulnerability remediation in the `handlebars` dependency.** This release updates `handlebars` to version 4.7.7, remediating the vulnerability.
- **Vulnerability remediation in the `ejs` dependency.** This release updates `bull-board` to 1.3.0, which includes version 3.1.6 of the `ejs` dependency, remediating the vulnerability.
- **Vulnerabilities remediation in the OpenSSL dependency.** These vulnerabilities are remediated for all images except `postgres` and `oauth2_proxy`, and resolves the following CVEs:
  - CVE-2021-3450
  - CVE-2021-3449

Comply 1.0.2

Released March 2021.

New in this release:

- **CIS assessor upgraded to 4.3.1.** Comply now uses a licensed version of the CIS assessor. To upgrade, see Upgrade the CIS assessor.
- **Windows Server 2016 STIG benchmark.** This new benchmark includes the Level 3 STIG Domain Controller profile.

Resolved in this release:

- **Activity feed empty.** Previously, the activity feed broke when the job had been purged in Puppet Enterprise (PE). This is now fixed.

- **TheQ logs not included.** TheQ logs are now included in the support bundle.
- **Large reports cannot be ingested.** Comply can now ingest XML files up to 32MB.
- **UI sending incorrect parameters.** This release fixes an issue where custom profile rules could not be updated.
- **Timeout prevents assessor download.** This release fixes an issue that prevented the assessor archive from downloading.
- **Custom profile ID not passing.** Comply now passes the custom profile ID in a scan task.
- **License uses incorrect casing**. This release fixes incorrect casing of the scarp response in the Comply license.

Security notice:

- **Vulnerability in lodash.** This release resolves the following risk vulnerabilities in the lodash library: CVE-2021-23337 and CVE-2020-28500.
- **Vulnerability in echo.** This release removes the echo dependency.
- **Vulnerability in i18next dependency.** This release resolves the vulnerability in the i18next dependency.
- **Vulnerability found in image.** This release resolves CVE-2021-23840.
- **Postgres 12.5 vulnerable.** The version of Postgres included in Comply has been upgraded to version 12.6 and resolves the following CVEs: CVE-2020-36221, CVE-2020-36222, CVE-2020-36223, CVE-2020-36224, CVE-2020-36225, CVE-2020-36226, CVE-2020-36227, CVE-2020-36228, CVE-2020-36229, CVE-2020-36230, CVE-2020-36221, CVE-2020-36222, CVE-2020-36223, CVE-2020-36224, CVE-2020-36225, CVE-2020-36226, CVE-2020-36227, CVE-2020-36228, CVE-2020-36229, and CVE-2020-36230

Comply 1.0.1

Released February 2021.

New in this release:

- **Preflight check for volume use**. This preflight check verifies the Ceph storage layer.
- **Preflight check to verify hostname is reachable**. This preflight check ensures that the Comply application can communicate with the configured hostname.
- **Support bundle analyzers**. Support bundles now include analyzers for preflight checks and issues with application components. Preflight checks also verify that schedulable CPU and memory capacity are available to perform upgrades.
- **Updated log levels**. A new configuration option in the KOTS admin allows you to modify Comply's debugging output.

Resolved in this release:

- **Pre-flight false positive**. The hostname preflight check no longer returns false positives.
- **Report files left in queue**. Report files are no longer left in the queue service filesystem.

Security notice:

- **Comply UI vulnerabilities**. This release fixes UI service vulnerabilities.
- **Queue service vulnerabilities**. This release fixes queue service vulnerabilities.
- **Postgres container base image issues.** This release updates the postgres container to fix the following security issues: CVE-2020-29361, CVE-2020-29362, and CVE-2020-29363.

Comply 1.0.0

Released December 2020.

Features in this release:

- **CIS scans.** Check the compliance of your nodes against CIS Benchmarks. For a list of supported operating systems, see system requirements.
- **Desired compliance.** Set a default benchmark and profile that you want your scans to be measured against.
- **Custom profiles.** Customize profiles to specify which rules you want visible in scan reports.

- **Compliance status.**The **Compliance dashboard** shows the compliance status of your nodes based on the latest scan results.
- **Node breakdown.** The **Node compliance** page shows an individual node's compliance status.
- **Rule breakdown.** The **Rules results** page shows the status of a rule on each node that is checked, why the rule is important, and specific operating system steps you can take to fix a rule that is failing scans.

Comply known issues

These are the known issues for the Comply 1.x release series.

Running scan tasks in Puppet Enterprise (PE)

Comply uses PE tasks to run compliance scans on nodes. While you can see the scan task in PE, we advise *against* running them from here as it can have unforeseen effects on both PE and Comply. Instead, run all CIS scans from Comply. You can view the results of the scan in both products.

Running scans on CentOS 7 with Comply 1.0.4

The CentOS 7 benchmark in Comply 1.0.4 has been updated to version 3.1.0. If you have already installed Comply and set desired compliance for your CentOS 7 nodes, you need to run following command on your `comply-scarpy` pod to update the benchmark version from 3.0.0 to 3.1.0:

```
kubectl exec --stdin --tty -n <namespace> $(kubectl get pods -n dio-comply
 | grep comply-scarpy | awk '{print $1}') -- /bin/scarp upgrade-assessor --
assessor_version '4.6.0'
```

This ensures Comply uses the latest CISCAT benchmark and profiles.

Beginner’s guide to Comply

Welcome to the Beginner’s guide to Comply! As a new user, you'll need to perform some initial installation and configuration tasks, and then we'll show you how to use the core features of Comply.

You're just a few steps away from enforcing compliant configurations across your infrastructure. Before you begin, we recommend familiarizing yourself with our terminology and Comply overview on page 4.

Step 1: Install and configure Comply

Use the main documentation to install and configure Comply. If you've already completed these steps, proceed to step 2.

- Install Puppet Application Manager (PAM)
- Set up Comply

Related concepts

Install PAM on page 41
You can install Puppet-supported Puppet Application Manager on a single node or in an HA configuration. Both online and offline install packages are available. You can also install it on an existing Kubernetes cluster.

Set up Comply on page 87
To start using Puppet Comply, you must complete the setup process, using both Comply and Puppet Enterprise (PE).

Step 2: Set desired compliance

Desired compliance is the benchmark and profile that you to assign to a particular node. It is what is scanned on that node by default. Most of the time, you only need to set this once for your nodes.

Based on fact information from PE, Comply can automatically assign an appropriate benchmark for each operating system, along with a Level 1 profile, to nodes that have not been set. This is the quickest way to get up and running with desired compliance. To manually choose your own benchmark and profiles, see Manually set desired compliance.

1. In Comply, click **Nodes**.

   Comply lists the nodes that have been classified with the `comply` class. If you do not see any nodes, ensure you have classified your nodes correctly.
2. In the message box that appears in the top right corner, click **Apply suggested profiles**.

   Comply automatically assigns profiles to all the nodes that have not already been set on your *current* page. To apply the suggested profile to all the nodes in your inventory, you must do this on every page.

> **Tip:** If you want to customize your scans to fit your organization's internally defined standards, see Creating custom profiles, which shows you how to exclude rules in a profile.

The ## sign in the **Profile assigned** column tells you that the desired compliance is set. You can view the node's information, including its assigned benchmark and profile, by clicking on the node. If you want to change a node's desired compliance, use the drop-down menu and click **Update**.

Step 3: Run a CIS scan

You are now ready to run a scan.

1. In Comply, click **Scan**.
2. In the **Benchmark** drop-down, select **Desired Compliance**.

   This scans each node with the profiles you assigned in the previous step.
3. Click **Next** to review the PE credentials and environment you want the scan to run on.
4. Click **Next** to see the nodes selected for scanning.

   To only scan a subset of nodes, deselect any that you do not want to include.
5. Click **Scan** and then **Start**.

   You'll be taken to the **Activity Feed**, which lists each scan. Scans are run as a task in PE. To see the details of the job, click on the job ID to be taken to PE.

> **Tip:** You can also run a scan by clicking the **Scan nodes** button at the top right corner on several pages. This option uses the nodes listed on the page you are currently viewing.

6. In Comply, navigate to the **Compliance dashboard** to see the results of your scan.

   See Viewing scan results for a description of the scan data.

Congratulations! You've completed the Beginner’s guide to Comply. You're now familiar with the core features and know how to run CIS scans with Comply.

Puppet Application Manager

Before you can begin using Puppet Comply, you must install Puppet Application Manager. Puppet Application Manager is an administrative console that provides tools for managing Comply and other Puppet applications.

Note: Puppet Application Manager was previously called the platform admin console in the Comply documentation.

What does Puppet Application Manager do?

The Puppet Application Manager installation process sets up a managed Kubernetes cluster (or, if you prefer, adds Puppet Application Manager to your existing cluster). Comply runs on this Kubernetes cluster, and Puppet Application Manager manages the cluster for you.

In the Puppet Application Manager UI, you can configure Comply, monitor the cluster's activity, upgrade to the latest version of the software, and back up your installation.

How do I use Puppet Application Manager to deploy Comply?

Once the cluster is ready, upload your Comply license and provide any needed configuration details about your installation in the Puppet Application Manager UI. You can then deploy the latest version of Comply with one click whenever you're ready.

Welcome to Puppet Application Manager (PAM)

Puppet Application Manager is an administrative console where you can install, access, and manage your Puppet applications. It is also where you can go to access upgrades to new Puppet applications releases.

Useful links:

Puppet Application Manager docs links	Other useful places
Before you install	**Docs for related Puppet products**
Release notes	Continuous Delivery for PE
System requirements	Comply
Install Puppet Application Manager	**Get support**
PAM standalone online installation on page 51	Support
PAM standalone offline installation on page 54	Upgrade your support plan
PAM HA online installation on page 44	**Share and contribute**
PAM HA offline installation on page 48	Engage with the Puppet community
Upgrading, disaster recovery, and troubleshooting	Puppet Forge
Upgrading PAM on a Puppet-supported cluster on page 67	Open source projects from Puppet on GitHub
Backing up PAM using snapshots on page 73	**External resources**
Disaster recovery with PAM on page 81	Getting started with Kubernetes Off-The-Shelf software (KOTS)
Troubleshooting PAM on page 82	

PAM UI

The Puppet Application Manager (PAM) UI provides administration functionality where you can access and manage your Puppet applications.

PAM console menu

Use the console menu at the top of the Puppet Application Manager UI to manage Puppet Application Manager itself. It has three tabs of interest to us:

- Use the **Dashboard** tab to:
  - Manage your applications
  - See version history
  - Set application configuration settings
  - Access support bundles for troubleshooting
  - Manage licenses
  - View files
  - Configure registry settings
- Use the **Cluster Management** tab to view current information on the nodes in your cluster. You can also use this tab to drain, and add nodes to your cluster.
- Use the **Snapshots** tab to create point-in-time backups of your deployment, which can be used to roll back to a previous state, or restore your installation into a new cluster for disaster recovery. For more information, see Backing up PAM using snapshots on page 73.

You can also use the console menu to **Add a new application** and to log out of Puppet Application Manager.

Application monitoring graphs

When you have Prometheus installed, the **Dashboard** tab has an **Application** sub-tab that provides several simplified graphs for tracking overall health of the system.

- **Node CPU Usage (%)** shows when hosts are getting overwhelmed (high % usage).
- **Node Memory Usage (%)** shows when hosts are reaching full memory capacity that may result in processes being killed due to out-of-memory errors.
- **Node Available Storage (%)** shows when hosts are running out of storage. At 15%, pods may start to be evicted or reads/writes on databases are paused until more storage is made available.
- **Volume Available Storage (%)** shows when application persistent volumes are getting full (low %) that may lead to problems with a particular application. Note that -

> **Note:** As of the 30 June 2021 Puppet Application Manager release, the monitoring/Prometheus-Kubernetes pods limit their storage use and are expected to never fall below 10% available storage.

Puppet Application Manager HA architectures include Prometheus and Grafana. Metrics about how the system is working are sent to Prometheus, and can be displayed with Grafana. Grafana credentials are printed during install, or can be retrieved later with the following command:

```
kubectl -n monitoring get secret grafana-admin -o go-template='{{index .data
 "admin-user"|base64decode}}:{{index .data "admin-password"|base64decode}}'
```

Related information

Backing up PAM using snapshots on page 73
Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

Release notes

PAM release notes

These are the new features, enhancements, resolved issues, and deprecations for Puppet Application Manager.

> **Restriction:** Because kURL does not support upgrading more than two Kubernetes minor release versions at once, if you're upgrading from an older version of PAM, you might need to follow a specific upgrade path to avoid failures. For example, PAM version 1.80.0 uses Kubernetes version 1.**21**.x, so you can upgrade up to PAM 1.91.3 (Kubernetes version 1.**23**.x), but not to PAM 1.94.0 (Kubernetes version 1.**24**.x). To determine the specific upgrade path for your installation, please check the table of Kubernetes versions for each version of PAM.

26 March 2024 (Puppet Application Manager 1.108.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

  - KOTS: 1.108.0
  - kURL: v2024.02.23-0
  - containerd: 1.6.28
  - Flannel: 0.24.2
  - Project Contour: 1.27.0
  - Velero: 1.12.3
  - Metrics Server: 0.6.4
  - ekco: 0.28.4
  - Prometheus: 0.71.2-56.6.0
  - OpenEBS: 3.10.0
  - MinIO: 2024-02-17T01-15-57Z

13 February 2024 (Puppet Application Manager 1.107.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

  - KOTS: 1.107.0
  - kURL: v2024.01.09-0
  - containerd: 1.6.26
  - Flannel: 0.24.0
  - Project Contour: 1.27.0
  - Velero: 1.12.2
  - ekco: 0.28.4
  - Prometheus: 0.70.0-55.0.0
  - OpenEBS: 3.10.0
  - MinIO: 2024-01-01T16-36-33Z
  - Rook: 1.12.8

7 November 2023 (Puppet Application Manager 1.103.3)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.28.2.

  > **Important upgrade information:** The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.27.6 before upgrading to version 1.28.2 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.
  >
  > Additionally, please keep in mind that kURL can only be upgraded two minor versions at a time on page 86.

- **Component upgrades to address security issues.** This version upgrades the following:

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

  - KOTS: 1.103.3
  - kURL: v2023.10.26-0
  - containerd: 1.6.24
  - Flannel: 0.22.3
  - Project Contour: 1.26.1
  - Registry: 2.8.3
  - Velero: 1.12.1
  - OpenEBS: 3.9.0
  - MinIO: 2023-10-16T04-13-43Z
  - Rook: 1.12.6

26 September 2023 (Puppet Application Manager 1.102.2)

New in this release:

- **Migrated from Weave to Flannel.** Flannel has replaced Weave as the Kubernetes CNI on Puppet-supported clusters, as Weave is no longer supported. The installation has additional interactive prompts to support this change.

  > **Important upgrade information:**
  >
  > - IPv6 and dual-stack networks are not supported on Flannel.
  > - Pod-to-pod networking now depends on UDP port 8472 being open instead of ports 6783 and 6784.

- **Added a host preflight.** Added a host preflight in the installer to stop installation if the installer detects the presence of a default REJECT rule in the FORWARD chain of iptables.

  > **Important upgrade information:** This is a known issue with the Flannel installation. To check for a REJECT rule in the FORWARD chain of iptables, run:
  >
  > ```
  > iptables -vL FORWARD
  > ```
  >
  > If there are any REJECT rules, those rules must be removed prior to the upgrade. They can be restored afterwards.

- **Component upgrades to address security issues.** This version upgrades the following:

  > **Note:** Before updating standalone installations, ensure there is at least 10GB of free space in `/var/openebs` to allow for migration of MinIO in this release.

  - KOTS: 1.102.2
  - kURL: v2023.09.15-0
  - containerd: 1.6.22
  - Weave: REMOVED
  - Flannel: 0.22.2
  - Project Contour: 1.25.2
  - Velero: 1.11.1
  - Kubernetes Metrics Server: 0.6.4
  - ekco: 0.28.3
  - Prometheus: 0.68.0-51.0.0
  - OpenEBS: 3.8.0
  - MinIO: 2023-09-04T19-57-37Z
  - Rook: 1.12.3

  > **Note:** If you are using the firewall module to manage your PAM install, you must update it to version 1.0.4 to support this PAM release.

18 July 2023 (Puppet Application Manager 1.100.3)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.26.6.

  > **Important upgrade information:** The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.25 before upgrading to version 1.26.6 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.
  >
  > Additionally, please keep in mind that kURL can only be upgraded two minor versions at a time on page 86.

- **Component upgrades to address security issues.** This version upgrades, adds, and removes the following:

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

  - KOTS: 1.100.3
  - kURL: v2023.06.27-0
  - Prometheus: 0.65.2-46.8.0
  - OpenEBS: 3.7.0
  - MinIO: 2023-06-19T19-52-50Z
  - Rook: 1.11.8

  > **Note:** If you are using the firewall module to manage your PAM install, you must update it to version 1.0.3 to support this PAM release.

8 June 2023 (Puppet Application Manager 1.99.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

  - KOTS: 1.99.0
  - kURL: v2023.05.22-0
  - containerd: 1.6.21
  - Weave: 2.8.1-20230417
  - Project Contour: 1.25.0
  - Registry: 2.8.2
  - Velero: 1.11.0
  - ekco: 0.27.1
  - Prometheus: 0.65.1-45.28.0
  - OpenEBS: 3.6.0
  - MinIO: 2023-05-18T00-05-36Z
  - Rook: 1.11.5
  - Goldpinger: 3.7.0-6.0.1

  > **Note:** For offline HA installs the Rook update in this release can cause significant downtime (around 4 hours) while downloading additional files. It is possible to do some of this prior to upgrading Puppet Application Manager from 1.97.0 to 1.99.0 to decrease the downtime.

25 April 2023 (Puppet Application Manager 1.97.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of OpenEBS to version 3.5.0, an upgrade of kURL to v2023.04.11-0, an upgrade of containerd to 1.6.20, an upgrade of Weave to version 2.8.1-20230324, an upgrade of Project Contour to version 1.24.3, an upgrade of ekco to 0.26.5, an upgrade of Velero to version 1.10.2, an upgrade of the Prometheus bundle to version 0.63.0-45.9.1, and upgrade of Kubernetes Metrics Server to version 0.6.3, an upgrade of KOTS to 1.97.0, an upgrade of MinIO to version 2023-03-24T21-41-23Z, and an upgrade of Goldpinger to 3.7.0-5.6.0.

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

Deprecated in this release:

- **`force-reapply-addons` flag.** Starting with Puppet Application Manager 1.97.0, the `force-reapply-addons` flag is deprecated and generates a warning on use. This flag is only required when upgrading to a Puppet Application Manager version prior to 1.97.0.

28 February 2023 (Puppet Application Manager 1.94.0)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.24.10.

  > **Important upgrade information:** The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.24.10 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.
  >
  > Additionally, because kURL can only be upgraded two minor versions at a time on page 86, if you're on PAM version 1.80.0 or earlier, you must upgrade to PAM 1.81.1 before upgrading to PAM 1.94.0.

- This release also includes component upgrades to address security issues and general bug fixes.

10 January 2023 (Puppet Application Manager 1.91.3)

New in this release:

- **Component upgrades to address security issues and support RHEL 8.7.** This version upgrades the following:

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

  - KOTS: 1.91.3
  - MinIO: 2022-10-20T00-55-09Z
  - OpenEBS: 3.3.0
  - Prometheus: 0.60.1-41.7.3
  - ekco: 0.26.1
  - Velero: 1.9.4
  - Project Contour: 1.23.1
  - kURL: v2022.12.12-0
  - Weave: 2.8.1-20221122
  - Goldpinger: 3.7.0-5.5.0

28 September 2022 (Puppet Application Manager 1.81.1)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.23.9.

  > **Important upgrade information:** The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.22 before upgrading to version 1.23.9. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.
  >
  > Additionally, because kURL can only be upgraded two minor versions at a time on page 86, if you're upgrading from PAM version 1.56.0 or earlier, you must upgrade to PAM 1.80.0 before upgrading to PAM 1.81.1.

  For legacy installations, Kubernetes remains on version 1.19.15. If you're not sure which installation type you're running, see How to determine your version of Puppet Application Manager.

16 August 2022 (Puppet Application Manager 1.80.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version upgrades containerd to 1.4.13, KOTS to 1.80.0, ekco to 0.19.6, and Goldpinger to 3.5.1-5.2.0.

Resolved in this release:

- Fixed an issue where legacy encryption keys didn't load properly during snapshot restores.

2 August 2022 (Puppet Application Manager 1.76.2)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of OpenEBS to version 3.2.0, an upgrade of Weave to version 2.8.1-20220720, an upgrade of Project Contour to version 1.21.1, and an upgrade of MinIO to version 2022-07-17T15-43-14Z.

  > **Note:** Before updating, ensure MinIO has 10GB of free space.

20 July 2022 (Puppet Application Manager 1.76.1)

New in this release:

- **Support for Red Hat Enterprise Linux version 8.6.** Beginning with version 1.76.1, PAM can be successfully installed on systems running Red Hat Enterprise Linux version 8.6.
- **More log data is now retained.** To ensure that you and our Support team have the data you need in debugging scenarios, the size of the pod logs has been increased from 10 files of 10MiB each to 10 files of 50MiB each. This change increases the storage used in `/var/log/pods` by 400MiB.
- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Velero to version 1.9.0 and an upgrade of the Prometheus bundle to version 0.57.0-36.2.0.
- **Other component upgrades.** This version also includes an upgrade of Registry to version 2.8.1 and an upgrade of MinIO to version 2022-07-06T20-29-49Z.

> **Note:** Before updating, ensure MinIO has 10GB of free space.

Resolved in this release:

- Velero pods no longer get stuck in a pending state when creating a snapshot to be saved to internal storage on a Puppet-supported cluster.

23 June 2022 (Puppet Application Manager 1.72.1)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of ekco to version 0.19.2 and an upgrade of kURL to v2022.06.17-0.

26 May 2022 (Puppet Application Manager 1.70.1)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Project Contour to version 1.21.0, an upgrade of Velero to version 1.8.1, and an upgrade of the Prometheus bundle to version 0.56.2-35.2.0.

Resolved in this release:

- Image garbage collection in Kubernetes installer-created clusters (embedded clusters) no longer removes images outside of the application's dedicated registry namespace.
- The **Deploy** button is now present in newly updated versions after the configuration is updated from the previously deployed version.
- Legends are now shown properly for the performance graphs on the dashboard.

12 April 2022 (Puppet Application Manager 1.68.0)

New in this release:

- **Install a specific version of an application.** When installing a Puppet application using the automated installation method, you now have the option to specify the application's version by passing the `--app-version-label=<version>` flag to the `kubectl kots install` command. For more information, go to Automate PAM and Puppet application online installations on page 56.
- **Status reporting improvements.** The status reporting tools can now detect when an application is being upgraded.
- **Component upgrades to address CVEs.** To address various CVEs in Envoy, this version includes an upgrade of Project Contour to version 1.20.1.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.68.0, which enables Kubernetes audit event logging by default and adds a 1 GB storage requirement for `/var/log/apiserver`.

Resolved in this release:

- During image garbage collection, images still in use by the cluster are no longer in danger of being deleted from the private registry in a Kubernetes installer-created cluster.

1 March 2022 (Puppet Application Manager 1.64.0)

Resolved in this release:

- Diffs are now shown correctly in the PAM UI.
- The OpenSSL package is no longer a prerequisite for successful installation on newer Red Hat Enterprise Linux 7 systems.
- You can now successfully install Puppet Application Manager on Red Hat Enterprise Linux 8 systems without the need to force-install the kurl-local audit-libs library.

17 February 2022 (Puppet Application Manager 1.62.0)

Important: Version 1.0.2 of the `puppetlabs/pam_firewall` module is now available. To avoid conflicts, upgrade the module **before** upgrading Puppet Application Manager to version 1.62.0.

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.21.8.

  Important upgrade information: The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.20 before upgrading to version 1.21.8. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

  For legacy installations (installed before May 2021), this version includes an upgrade of Kubernetes to version 1.19.15.

  Tip: See How to determine your version of Puppet Application Manager if you're not sure which installation type you're running.
- **Prometheus enabled on standalone architecture.** Beginning with version 1.62.0 Prometheus is enabled by default on all new and existing standalone Puppet Application Manager installations. Prometheus requires an additional 350m CPU and 500MiB of memory, so ensure your system is properly sized before upgrading. Prometheus is an optional component; if you need to disable it to conserve resources, see Optional components on page 85.
- **Automatic certificate rotation.** By default, the self-signed certificates used by Project Contour and Envoy expire after one year. This version includes an update that auto-rotates those certificates before they expire.
- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of containerd to version 1.4.12.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.62.0.

Deprecated in this release:

- **Legacy architecture.** The legacy architecture, which was the version of Puppet Application Manager available for installation prior to May 2021, is now deprecated. (See How to determine your version of Puppet Application Manager if you need to confirm whether you're running the legacy architecture.) The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. Puppet will continue to update legacy architecture components other than Kubernetes until 30 June 2022. If security advisories against Kubernetes 1.19 arise, the remediation path is to migrate to one of the newer architectures by following the instructions in Migrating PAM data to a new system on page 75.

  Important: Before beginning the migration process from a legacy deployment you must upgrade to PAM version 1.62.0 with the `force-reapply-addons` flag included in the upgrade command. Find upgrade instructions at PAM legacy upgrades on page 70 and PAM offline legacy upgrades on page 70.

30 November 2021 (Puppet Application Manager 1.56.0)

This release includes an upgrade of KOTS to version 1.56.0, which adds the following improvements:

- **Improved support bundles:** Adds an option to upload a support bundle directly from Puppet Application Manager.
- **Improved troubleshooting**: Adds detailed information on failing pods to the **Troubleshoot** tab.

6 October 2021 (Puppet Application Manager 1.52.1)

New in this release:

- **Improved statuses**. More granular status levels are now available from the **Application** tab.
- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Kubernetes to 1.19.15.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.52.1.

Resolved in this release:

- Generating a support bundle no longer results in unusually high memory use.
- Preflight check logs post to info level for progress messages and to error level for error messages.

25 August 2021 (Puppet Application Manager 1.49.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Kubernetes to 1.19.13, an upgrade of Project Contour to 1.18.0, and an upgrade of Velero to 1.6.2.
- **Goldpinger.** High availability architectures now include Goldpinger, which aids the debugging of network issues.
- **containerd upgrade.** This version includes an upgrade of containerd to version 1.4.6, and removes the need to use the `force-reapply-addons` option when upgrading.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.49.0, an upgrade of ekco to 0.11.0, an upgrade of Prometheus to 0.49.0, and an upgrade of Rook to 1.5.12.

30 June 2021 (Puppet Application Manager 1.44.1)

New in this release:

- **Certificate auto-rotation for standalone architecture.** Certificates are now automatically rotated for the Kubernetes API and Puppet Application Manager UI in the standalone architecture. With this change, certificate auto-rotation is now supported in all Puppet Application Manager architectures.
- **Rook upgrades.** This version includes an upgrade of Rook in the high availability architecture to 1.5.11 and the version of Rook in the legacy architecture to 1.0.4-14.2.21. These upgrades address a vulnerability in Ceph components (CVE-2021-20288).
- **Prometheus upgrade.** This version includes an upgrade of Prometheus in the high availability and legacy architectures to 0.48.1. Additionally, Prometheus disk usage is now limited in order to preserve the storage space required for the usage charts on the **Application** tab.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.44.1, an upgrade of Project Contour to version 1.15.1, and an upgrade of Weave to version 2.8.1.

Resolved in this release:

- Snapshots can now successfully use the **Other S3-Compatible Storage** option as the storage destination.

  To apply this update, add the `force-reapply-addons` option during upgrade. For example:

  ```
  curl <url> | bash -s force-reapply-addons
  ```

26 May 2021

New in this release:

- **runC**. The version of runC has been upgraded to v1.0.0-rc95 to address CVE-2021-30465.

Known issues in this release:

- Running the KOTS installer with the `airgap` and `kurl-registry-ip` flags results in an error.

  As a workaround (if you do not have any applications already installed in the cluster), delete the registry service, recreate the registry service IP and then re-run the installation script with the `kurl-registry-ip` flag.

10 May 2021 (Puppet Application Manager 1.40.0)

New in this release:

- Distinct architectures for standalone and high availability deployments of the Puppet Application Manager platform. Standalone supports lower system requirements and resolves inherent flaws in using Ceph on a single node. High availability uses an updated version of Rook for faster, more reliable distributed storage.

> **Note:** It is not possible currently to upgrade to these architectures from existing installations. However, migrating applications between them is on the roadmap for a future release.

- The previous architecture is maintained as the legacy configuration. This version includes an upgrade of Kubernetes to 1.19.10; this upgrade process upgrades through Kubernetes 1.18, and happens on all nodes. It can take ~1 hour to do for a 3-node cluster, and requires confirmations during that period. It also includes an upgrade of Project Contour to version 1.14.1, adds Metrics Server 0.4.1, an upgrade of ekco to 0.10.1, and an upgrade of Prometheus to 2.26.0.

  For more information on legacy upgrades, see PAM legacy upgrades on page 70.

15 April 2021 (Puppet Application Manager 1.38.0)

New in this release:

- **Snapshots.** Puppet Application Manager now supports full (instance-level) snapshots, which can be used for application rollbacks and disaster recovery. For more information, see **Backing up Puppet Application Manager using snapshots**.
- **Component upgrades.** This version includes an upgrade of KOTS to version 1.38.0.

17 February 2021 (Puppet Application Manager 1.29.3)

New in this release:

- **Support for Ubuntu 20.04.** You can now run Puppet Application Manager on Ubuntu 20.04.
- **Component upgrades.** This version includes an upgrade of Prometheus to version 2.22.1 and Prometheus Operator to version 0.44.1, an upgrade of KOTS to version 1.29.3, an upgrade of Project Contour to version 1.12.0, and an upgrade of ekco to version 0.10.0.

3 February 2021 (Puppet Application Manager 1.29.2)

New in this release:

- **Component upgrades.** This version includes an upgrade of KOTS to version 1.29.2, an upgrade of Project Contour to version 1.11.0, and an upgrade of `containerd` to version 1.4.3.

Resolved in this release:

- During their initial preflight checks, new installations now pull images successfully and no longer report a `Failed to pull image` error.

7 December 2020

New in this release:

- **Support for Red Hat Enterprise Linux (RHEL) 8 and CentOS 8.** You can now run Puppet Application Manager on RHEL version 8 and CentOS version 8. To support this change, `containerd` is now used independently of Docker during the installation process.

- **Component upgrades.** This version includes an upgrade of Kubernetes to version 1.17.13.

Related information

Upgrading PAM on a Puppet-supported cluster on page 67
Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

Backing up PAM using snapshots on page 73
Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

Known issues

These are the known issues for Puppet Application Manager (PAM).

Restarting a PAM node on v1.103.3 and v1.107.0 does not successfully bring back up all pods

A known Kubernetes issue impacts PAM versions v1.103.3 and v1.107.0. Rebooting a node can result in pods incorrectly changing phases, causing them to remain at states such as 0/1 Completed and 0/1 Error. This can lead to pod crashloops and application outages. To work around this issue, delete the errant pods, which forces them to restart. Check the pods afterwards to ensure they come up as “Running”.

PAM versions 1.72.1 and older cannot be installed on RHEL 8.6+ systems

A known issue in kURL prevents Puppet Application Manager versions 1.72.1 and older from successfully installing on Red Hat Enterprise Linux (RHEL) version 8.6 and newer versions. To work around this issue, install or upgrade to Puppet Application Manager version 1.76.1 or a newer version, which support RHEL version 8.6.

Velero fails if network file system (NFS) snapshot storage is misconfigured

In Puppet Application Manager version 1.64.0 and newer versions, changes to the configuration of snapshot storage on a network file system (NFS) is appended to Velero containers, rather than replaced. This means that if NFS snapshot storage is misconfigured, attempts to fix the configuration do not correct the problem. This issue manifests as a failure of Velero to start up.

OpenSSL package required for newer RHEL 7 systems with PAM 1.62.0

Attempts to install Puppet Application Manager version 1.62.0 or older on newer Red Hat Enterprise Linux (RHEL) 7 systems fail unless the OpenSSL package is present on the system before installation. To work around this issue, run `yum install openssl` and then re-run the PAM installation script.

Package updates with yum or DNF fail after upgrading PAM

If you are unable to run `yum update` or `dnf upgrade` after a PAM upgrade, run one of the following commands to clean up a temporary module added by PAM:

```
yum module reset kurl.local
```

or

```
dnf module reset kurl.local
```

Architecture overview

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

PAM can run on Puppet-supported or customer-supported Kubernetes clusters. Due to potential variations in the architecture of customer-supported clusters, the architecture overview provided on this page assumes PAM is running on Puppet-supported clusters. For more information on installing on a customer-supported Kubernetes cluster, see Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 42.

Terminology

Throughout this documentation, we use a few terms to describe different roles nodes can take:

- **Primary** - A primary node runs core Kubernetes components (referred to as the Kubernetes control plane) as well as application workloads. At least three primaries are required to support high availability for Puppet Application Manager. These are also sometimes referred to as *masters*.
- **Secondary** - A secondary node runs application workloads. These are also sometimes referred to as *workers*.

Puppet Application Manager is built on the KOTS (Kubernetes off-the-Shelf) project, and we occasionally use its CLI tools (`kubectl`, `kots`) to manage the installation.

Standalone architecture

Standalone is optimized for limited resources, storing data directly on disk. If you need to remove optional components like Prometheus and Grafana to decrease resource utilization, see Optional components on page 85. While additional compute capacity can be added through secondary nodes, this does not provide increased resilience as data is only stored on the node where a component service runs.

For information on migrating data from standalone to HA deployments, see Migrating data between two systems with different architectures on page 80.

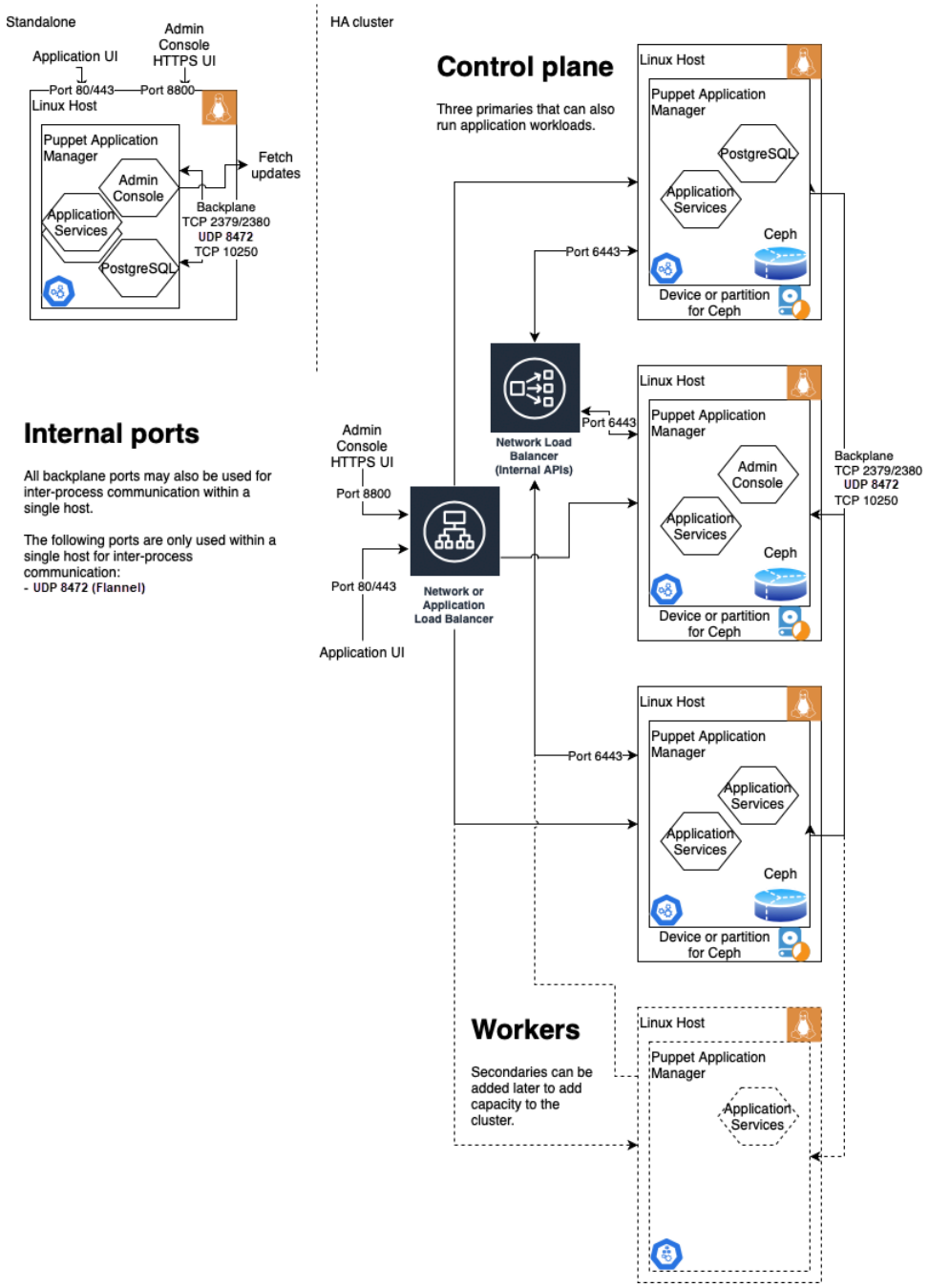
HA architecture

A high availability (HA) architecture provides high availability for scheduling application services during failure and uses Ceph for distributed storage in case of node failure. Individual applications may still experience some loss of availability (up to 10 minutes) if individual services do not have replicas and need to be rescheduled. For more information, see Reduce recovery time when a node fails on page 84. An HA implementation requires a cluster of three primary nodes. Additional compute capacity can be added through secondary nodes.

The HA architecture installs Prometheus and Alertmanager. These are used to provide system monitoring in the Puppet Application Manager UI. Prometheus and Alertmanager are unauthenticated on ports 30900 and 30903, and you are recommended to control access to these ports via firewall rules. For information on how to remove Prometheus and Alertmanager, see Optional components on page 85.

Puppet Application Manager architectures

The following diagram and lists outline some of the core components involved in standalone and HA architectures and how they communicate. For a detailed list of ports used by Puppet Application Manager, refer to the **Cluster port requirements** sections of the PAM system requirements on page 27. For firewall information, refer to Web URL and port requirements for firewalls on page 39.



Standalone architecture

Puppet Application Manager

Lives on a cluster within a Linux host.

The PAM application includes the admin console, application services, and PostgreSQL.

PAM communicates out of the Linux host to fetch updates.

UI ports

The application UI communicates on 80/443 to the Linux host.

The admin console HTTPS UI communicates on 8800 to the Linux host.

Backplane and internal ports

Backplane ports include 8472 (UDP) and 10250 (TCP).

Backplane ports can also be used within a single host for inter-process communication.

These ports are only used within a single host for inter-process communication (Flannel): 8472 (UDP)

Additional default ports

30900: Prometheus UI

30902: Grafana UI

30903: Alertmanager UI

HA cluster architecture

Control plane (primaries)

Multiple primaries that can also run application workloads.

Structured as clusters within Linux hosts with a device or partition for Ceph.

Each primary hosts PAM and can run application services in addition to supporting either PostgreSQL or the admin console.

Workers (secondaries)

Can be added later to add capacity for running application workloads.

Structured as clusters within Linux hosts.

Network or Application Balancer

The balancer communicates out to the control plane (primaries) and workers (secondaries).

Receives admin console HTTPS UI communication over 8800.

Receives application UI communication over 80/443.

Network load balancer internal APIs communicate with primaries and secondaries over 6443.

To learn about setting up health checks for your load balancer, go to Load balancer health checks on page 66.

Backplane and internal ports

Backplane ports include 2379/2380 (TCP), 8472 (UDP), and 10250 (TCP).

Backplane ports can also be used within a single host for inter-process communication.

These ports are only used within a single host for inter-process communication (Flannel): 8472 (UDP)

Additional default ports

30900: Prometheus UI

30902: Grafana UI

30903: Alertmanager UI

UNSUPPORTED: Legacy architecture

Note: The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. The legacy architecture reached the end of its support lifecycle on **30 June 2022**, and Puppet no longer updates legacy architecture components.

The Puppet Application Manager legacy architecture reflects an older configuration that used Ceph 1.0 which hosted data directly on the file system. Installing the legacy architecture is no longer supported.

For information on upgrading to a newer version of the legacy architecture, see PAM legacy upgrades on page 70 and PAM offline legacy upgrades on page 70.

For information on migrating data from a legacy architecture to a standalone or HA architecture, go to our Support Knowledge Base instructions:

- Migrate to a supported PAM architecture for Continuous Delivery for PE
- Migrate to a supported PAM architecture for Comply

Related information

Reduce recovery time when a node fails on page 84
If a node running a non-replicated service like PostgreSQL fails, expect some service downtime.

Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 42
Use these instructions to install Puppet Application Manager and any Puppet applications on an existing Kubernetes cluster.

PAM legacy upgrades on page 70
The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM).

PAM offline legacy upgrades on page 70
The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM) on offline nodes.

Troubleshooting PAM on page 82
Use this guide to troubleshoot issues with your Puppet Application Manager installation.

PAM system requirements

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Customer-supported cluster hardware requirements

The following Kubernetes distributions are supported:

- Google Kubernetes Engine
- AWS Elastic Kubernetes Service

If you use a different distribution, contact Puppet Support for more information on compatibility with PAM.

Application requirements:

Application	CPU	Memory	Storage	Ports
Continuous Delivery for Puppet Enterprise (PE)	3 CPU	8 GB	280 GB	Ingress, NodePort 8000 **Note:** NodePort is configurable
Puppet Comply®	7 CPU	7 GB	35 GB	Ingress, NodePort 30303 **Note:** NodePort is configurable

Make sure that your Kubernetes cluster meets the minimum requirements:

- Kubernetes version 1.24-1.26.
- A default storage class that can be used for relocatable storage.
- A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
- We currently test and support Google Kubernetes Engine (GKE) clusters.

Cluster ports: In addition to the NodePorts used by your Puppet applications, make sure that TCP port 443 is open for your ingress controller.

Puppet-supported HA cluster hardware requirements

A high availability (HA) configuration uses multiple servers to provide availability in the event of a server failure. A majority of servers must be available to preserve service availability. Below are suggested configurations for each application.

Continuous Delivery for Puppet Enterprise (PE)

Three servers (referred to as primaries during installation) with the following minimum requirements:

CPU	Memory	Storage	Open ports
6 CPU	10 GB	100 GB on an unformatted storage device. 1 GB for `/var/log/apiserver` for Kubernetes audit logs. An additional 140 GB for `/var/lib`. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided: • 2 GB for `/var/lib/etcd` • 10 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd` **Note:** The storage backend prefers the file system inhabited by `/var/lib/rook` to remain below 70% utilization. SSDs (or similarly low-latency storage) are recommended for `/var/lib/etcd` and `/var/lib/rook`.	**TCP**: 80, 443, 2379, 2380, 6443, 8000, 8800, and 10250 **UDP**: 8472

Puppet Comply

Three servers (referred to as primaries during installation) with the following minimum requirements:

CPU	Memory	Storage	Open ports
7 CPU	10 GB	100 GB on an unformatted storage device. 1 GB for `/var/log/apiserver` for Kubernetes audit logs. An additional 140 GB for `/var/lib`. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided: • 2 GB for `/var/lib/etcd` • 10 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd` **Note:** The storage backend prefers the file system inhabited by `/var/lib/rook` to remain below 70% utilization. SSDs (or similarly low-latency storage) are recommended for `/var/lib/etcd` and `/var/lib/rook`.	**TCP**: 80, 443, 2379, 2380, 6443, 8800, 10250, and 30303 **UDP**: 8472

Continuous Delivery for Puppet Enterprise (PE) and Puppet Comply

Three servers (referred to as primaries during installation) with the following minimum requirements:

CPU	Memory	Storage	Open ports
8 CPU	13 GB	150 GB on an unformatted storage device. 1 GB for `/var/log/apiserver` for Kubernetes audit logs. An additional 140 GB for `/var/lib`. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided: • 2 GB for `/var/lib/etcd` • 10 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd` **Note:** The storage backend prefers the file system inhabited by `/var/lib/rook` to remain below 70% utilization. SSDs (or similarly low-latency storage) are recommended for `/var/lib/etcd` and `/var/lib/rook`.	**TCP**: 80, 443, 2379, 2380, 6443, 8000, 8800, 10250, and 30303 **UDP**: 8472

For a detailed example of an HA configuration running Continuous Delivery for PE and Puppet Comply, see Example of an HA cluster that supports CDPE and Comply.

Networking requirements

Gigabit Ethernet (1GbE) and a latency of less than 10 milliseconds (ms) between cluster members is sufficient for most deployments. For more information on networking for specific Puppet Application Manager components, see the documentation for Ceph, and etcd.

Cluster port requirements

Puppet Application Manager (PAM) uses the following ports in an HA cluster architecture:

Category	Port	Protocol	Purpose	Source
Puppet application ports	443	TCP	Web UI Relies on Server Name Indication to route requests to the application.	Browser
Continuous Delivery for Puppet Enterprise (PE) ports	8000	TCP	Webhook service	Source control

Category	Port	Protocol	Purpose	Source
Puppet Comply ports	30303	TCP	Communication with Puppet Enterprise (PE)	PE instance
Platform ports	2379, 2380	TCP	High availability (HA) communication Only needs to be open between the cluster's primary nodes.	etcd on the Kubernetes host.
	6443	TCP	Kubernetes API Might be useful to expose to workstations.	Admin workstation
	8472	UDP	Kubernetes networking - Flannel	Kubernetes host
	8800	TCP	PAM	Admin browser
	9001	TCP	Internal registry in offline installs only. Requires configuring an Ingress to use this port.	Kubernetes host
	9090	TCP	Rook CSI RBD Plugin Metrics	Kubernetes host
	10250	TCP	Kubernetes cluster management Only communicates in one direction, from a primary to other primaries and secondaries.	Kubernetes host

Additionally, these ports are configured by default: 30900 (Prometheus UI), 30902 (Grafana UI), and 30903 (Alertmanager UI)

For Kubernetes-specific information, refer to Networking Requirements in the Kurl documentation.

IP address range requirements

Important: Puppet Application Manager must be installed on nodes with static IP assignments because IP addresses cannot be changed after installation.

Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See Resolve IP address range conflicts for instructions.

Note: The minimum size for CIDR blocks used by PAM are:

- `/23` for pod and service CIDRs
- Default of `/22` is recommended to support future expansion

Antivirus and antimalware considerations

Antivirus and antimalware software can impact PAM and its applications or prevent them from functioning properly.

To avoid issues, exclude the following directories from antivirus and antimalware tools that scan disk write operations:

- `/var/lib/rook`
- `/var/lib/kubelet`
- `/var/lib/containerd`

Firewall modules

If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

If you've already installed PAM, apply the `pam_firewall` module and then restart the `kube-proxy` service to recreate its iptables rules by running the following on a primary:

```
systemctl restart kubelet
                    kubectl -n kube-system delete pod -l k8s-app=kube-proxy
                    kubectl -n kube-flannel delete pod -l app=flannel
```

For more information, see the PAM firewall module.

Supported operating systems

Puppet Application Manager and the applications it supports can be installed on these operating systems:

Operating system	Supported versions
Amazon Linux	2
CentOS	7.4, 7.5, 7.6, 7.7, 7.8, 7.9 8.0, 8.1, 8.2, 8.3, 8.4
Oracle Linux	7.4, 7.5, 7.6, 7.7, 7.8, 7.9 8.0, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8
Red Hat Enterprise Linux (RHEL)	7.4, 7.5, 7.6, 7.7, 7.8, 7.9 8.0, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8 9.0, 9.1, 9.2

Operating system	Supported versions
Rocky Linux	9.0, 9.1, 9.2
Ubuntu (General availability kernels)	18.04 20.04 22.04

Puppet-supported standalone hardware requirements

Here are the suggested configurations for standalone installations.

Continuous Delivery for Puppet Enterprise (PE)

CPU	Memory	Storage	Open ports
4 CPU	8 GB	220 GB for `/var/lib` and `/var/openebs`This is primarily divided among: • 2 GB for `/var/lib/etcd` • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd` • 100 GB for `/var/openebs`	**TCP**: 80, 443, 2379, 2380, 6443, 8000, 8800, and 10250 **UDP**: 8472

Puppet Comply

CPU	Memory	Storage	Open ports
7 CPU	7 GB	220 GB for `/var/lib` and `/var/openebs`This is primarily divided among: • 2 GB for `/var/lib/etcd` • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd` • 100 GB for `/var/openebs`	**TCP**: 80, 443, 2379, 2380, 6443, 8800, 10250, and 30303 **UDP**: 8472

Cluster port requirements

Puppet Application Manager (PAM) uses the following ports in a standalone architecture:

Category	Port	Protocol	Purpose	Source
Puppet application ports	442	TCP	Web UI Relies on Server Name Indication to route requests to the application.	Browser
Continuous Delivery for Puppet Enterprise (PE) ports	8000	TCP	Webhook service	Source control
Puppet Comply ports	30303	TCP	Communication with Puppet Enterprise	PE instance
Platform ports	6443	TCP	Kubernetes API Might be useful to expose to workstations.	Admin workstation
	8472	UDP	Kubernetes networking - Flannel	Kubernetes host
	8800	TCP	PAM	Admin browser
	9001	TCP	Internal registry in offline installs only. Requires configuring an Ingress to use this port.	Kubernetes host
	10250	TCP	Kubernetes cluster management Only communicates in one direction, from a primary to other primaries and secondaries.	Kubernetes host

Additionally, these ports are configured by default: 30900 (Prometheus UI), 30902 (Grafana UI), and 30903 (Alertmanager UI)

For Kubernetes-specific information, refer to Networking Requirements in the Kurl documentation.

IP address range requirements

Important: Puppet Application Manager must be installed on nodes with static IP assignments because IP addresses cannot be changed after installation.

Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See Resolve IP address range conflicts for instructions.

Note: The minimum size for CIDR blocks used by PAM are:

- `/24` for pod and service CIDRs
- Default of `/22` is recommended to support future expansion

Antivirus and antimalware considerations

Antivirus and antimalware software can impact PAM and its applications or prevent them from functioning properly.

To avoid issues, exclude the following directories from antivirus and antimalware tools that scan disk write operations:

- `/var/openebs`
- `/var/lib/kubelet`
- `/var/lib/containerd`

Firewall modules

If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

If you've already installed PAM, apply the `pam_firewall` module and then restart the `kube-proxy` service to recreate its iptables rules by running the following on a primary:

```
systemctl restart kubelet
                    kubectl -n kube-system delete pod -l k8s-app=kube-proxy
                    kubectl -n kube-flannel delete pod -l app=flannel
```

For more information, see the PAM firewall module.

Detailed hardware requirements

For additional compute capacity, you can horizontally scale HA and standalone architectures by adding secondary nodes. During installation, only add secondaries after setting up all primaries.

You can add secondaries to HA and standalone architectures; however in standalone architectures, secondaries do not increase availability of the application, and data storage services are pinned to the host they start on and cannot be moved.

Here are the baseline requirements to run cluster services on primaries and secondaries. Any Puppet applications require additional resources on top of these requirements.

Node type	CPU	Memory	Storage	Open ports
Primary	4 CPU	7 GB	At least 50 GB on an unformatted storage device in addition to application-specific storage (below) for the Ceph storage backend. This can be satisfied by multiple devices if more storage is needed later, but should be balanced across primaries. 1 GB for `/var/log/apiserver` for Kubernetes audit logs. An additional 140 GB for `/var/lib`. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided: • 2 GB for `/var/lib/etcd` • 10 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd` **Note:** Ceph storage backend prefers the file system inhabited by `/var/lib/rook` to remain below 70% utilization. SSDs (or similarly low-latency storage) are recommended for `/var/lib/etcd` and `/var/lib/rook`.	**TCP**: 80, 443, 2379,2380, 6443, 8800, and 10250 **UDP**: 8472
Secondary	1 CPU	1.5 GB	1 GB for `/var/log/apiserver` for Kubernetes audit logs. 120 GB for `/var/lib`. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided: • 32 GB for `/var/lib/kubelet` • 80 GB for `/var/lib/containerd`	

Applications are composed of multiple smaller services, so you can divide CPU and memory requirements across multiple servers. The listed ports can be accessed from all primaries and secondaries, but only need to be exposed on nodes you include in your load balancer. Apply application-specific storage to all primary nodes.

Application-specific requirements:

Application	**CPU**	**Memory**	**Storage**	**Ports**
Continuous Delivery for Puppet Enterprise (PE)	3 CPU	8 GB	50 GB	80, 443, 8000
Puppet Comply	7 CPU	7 GB	50 GB	80, 443, 30303

The minimum recommended size for a secondary node is 4 CPU and 8 GB of memory to allow some scheduling flexibility for individual services.

Example of an HA cluster capable of running Continuous Delivery for PE and Comply

An HA cluster capable of running both Continuous Delivery for Puppet Enterprise (PE) and Puppet Comply requires 10 CPU and 15 GB of application-specific memory in addition to per-node baselines. You can create a cluster from 4 CPU, 8 GB nodes. Each primary uses all CPU and 7 GB of memory for cluster services, providing 0 CPU and 1 GB of memory for application workloads; each secondary uses 1 CPU and 1.5 GB of memory for cluster services, providing 3 CPU and 6.5 GB of memory for application workloads. Create the cluster as follows:

- Three primaries provide an excess of 3 GB of memory for application workloads. Each primary must have 150 GB of storage in an unformatted, unpartitioned storage device for Ceph and 140 GB of storage for `/var/lib`.
- Three secondaries provide an excess of 9 CPU and 19.5 GB of memory for application workloads. Each secondary must have 120 GB of storage for `/var/lib`.

This diagram illustrates the suggested three-node configuration for a cluster capable of running Continuous Delivery for Puppet Enterprise (PE) and Puppet

Web URL and port requirements for firewalls

Puppet Application Manager interacts with external web URLs for a variety of installation, configuration, upgrade, and deployment tasks. Puppet Application Manager uses the following web URLs for internal and outbound network traffic.

Category	URLs
Puppet Application Manager and platform	• get.replicated.com • registry.replicated.com • proxy.replicated.com • api.replicated.com • k8s.kurl.sh • kurl-sh.s3.amazonaws.com • replicated.app • registry-data.replicated.com
Container registries	• gcr.io • docker.io • index.docker.io • registry-1.docker.io • auth.docker.io • production.cloudflare.docker.com • quay.io
Puppet Enterprise	• pup.pt • forgeapi.puppet.com • pm.puppetlabs.com • amazonaws.com • s3.amazonaws.com • rubygems.org

For information about containers and firewalls, refer to the Networking Requirements in the Kurl documentation.

Firewall modules

If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

If you've already installed PAM, apply the `pam_firewall` module and then restart the `kube-proxy` service to recreate its iptables rules by running the following on a primary:

```
systemctl restart kubelet
                    kubectl -n kube-system delete pod -l k8s-app=kube-proxy
                    kubectl -n kube-flannel delete pod -l app=flannel
```

For more information, see the PAM firewall module.

Supported browsers

The following browsers are supported for use with the Puppet Application Manager UI:

Browser	Supported versions
Google Chrome	Current version as of release
Mozilla Firefox	Current version as of release
Microsoft Edge	Current version as of release
Apple Safari	Current version as of release

Component versions in PAM releases

These tables show the versions of components included in recent Puppet Application Manager (PAM) releases.

Component	PAM 1.108.0	PAM 1.107.0	PAM 1.103.3	PAM 1.102.2	PAM 1.100.3	PAM 1.99.0	PAM 1.97.0	PAM 1.94.0	PAM 1.91.3	PAM 1.81.1	PAM 1.80.0	PAM 1.76.2	PAM 1.76.1	PAM 1.72.1	PAM 1.70.1	PAM 1.68.0
Kubernetes	1.28.2	1.28.2	1.28.2	1.26.6	1.26.6	1.24.10	1.24.10	1.24.10	1.23.9	1.23.9	1.21.8	1.21.8	1.21.8	1.21.8	1.21.8	1.21.8
KOTS	1.108.0	1.107.0	1.103.3	1.102.2	1.100.3	1.99.0	1.97.0	1.94.0	1.91.3	1.81.1	1.80.0	1.76.1	1.76.1	1.72.1	1.70.1	1.68.0
kURL	v2024.02.24-0	v2024.02.21-0	v2023.10.26-0	v2023.09.15-0	v2023.06.27-0	v2023.05.22-0	v2023.04.13-0	v2023.02.06-1	v2022.08.22-0	v2022.08.08-0	v2022.07.20-0	v2022.07.15-0	v2022.06.27-0	v2022.03.29-0	v2022.03.04.08-0	
Weave	N/A	N/A	N/A	REMOVED	2.8.1-20230420	2.8.1-20230420	2.8.1-20230324	2.8.1-20230124	2.8.1-20221220	2.8.1-20220720	2.8.1-20220720	2.8.1-20220720	2.8.1	2.8.1	2.8.1	2.8.1
Flannel	0.24.2	0.24.0	0.22.3	0.22.2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Project Contour	1.27.0	1.27.0	1.26.1	1.25.2	1.25.0	1.25.0	1.24.3	1.24.0	1.23.1	1.22.0	1.21.1	1.21.1	1.21.0	1.21.0	1.21.0	1.20.1
Registry	2.8.3	2.8.3	2.8.3	2.8.2	2.8.2	2.8.2	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.7.1	2.7.1	2.7.1
Prometheus bundle	0.71.2-67.0.0	0.70.0-58.0.0	0.69.0-55.0.0	0.65.2-46.0.0	0.65.2-46.0.0	0.62.0-45.0.0	0.59.1-44.0.0	0.59.1-44.0.0	0.57.0-39.1.1	0.57.0-39.1.1	0.57.0-39.1.1	0.57.0-39.1.1	0.57.0-39.1.1	0.50.2-35.0.0	0.50.2-35.0.0	0.50.2-35.0.0-17.1.1
containerd	1.6.28	1.6.26	1.6.24	1.6.22	1.6.21	1.6.21	1.6.20	1.5.11	1.4.13	1.4.13	1.4.13	1.4.12	1.4.12	1.4.12	1.4.12	1.4.12
Velero	1.12.3	1.12.2	1.12.1	1.11.1	1.11.0	1.11.0	1.10.2	1.9.5	1.9.4	1.9.1	1.9.0	1.9.0	1.9.0	1.8.1	1.8.1	1.6.2
ekco	0.28.4	0.28.4	0.28.3	0.28.3	0.27.1	0.27.1	0.26.5	0.26.3	0.26.1	0.20.0	0.19.6	0.19.2	0.19.2	0.19.2	0.16.0	0.16.0
Kubernetes Metrics Server	0.6.4	0.6.4	0.6.4	0.6.4	0.6.3	0.6.3	0.6.3	0.6.2	0.4.1	0.4.1	0.4.1	0.4.1	0.4.1	0.4.1	0.4.1	0.4.1

Component	PAM 1.108.0	PAM 1.107.0	PAM 1.103.3	PAM 1.102.2	PAM 1.100.3	PAM 1.99.0	PAM 1.97.0	PAM 1.94.0	PAM 1.91.3	PAM 1.81.1	PAM 1.80.0	PAM 1.76.2	PAM 1.76.1	PAM 1.72.1	PAM 1.70.1	PAM 1.68.0
Rook *(HA only)*	1.12.8	1.12.8	1.12.6	1.12.3	1.11.8	1.11.5	1.5.12	1.5.12	1.5.12	1.5.12	1.5.12	1.5.12	1.5.12	1.5.12	1.5.12	1.5.12
MinIO *(Standalone only)*	2024-02-24T01-33-52Z	2024-01-31T20-47-36Z	2023-10-25T06-16-23Z	2023-10-16T06-09-23Z	2023-09-07T07-13-43Z	2023-08-23T10-26-07Z	2023-06-19T23-11-33Z	2023-05-18T00-05-36Z	2023-04-07T05-37-25Z	2022-12-12T19-27-27Z	2022-11-08T05-27-05Z	2022-10-29T00-49-07Z	2022-07-26T19-43-11Z	2022-06-11T19-55-32Z	2022-05-26T05-51-49Z	2022-05-02T01-58-27Z
OpenEBS *(Standalone only)*	3.10.0	3.10.0	3.9.0	3.8.0	3.7.0	3.6.0	3.5.0	3.3.0	3.3.0	3.2.0	3.2.0	3.2.0	2.6.0	2.6.0	2.6.0	2.6.0

Looking up component versions

To view a list of the component versions included in your current version of PAM, run the following:

```
kubectl get installer -o jsonpath="{.items[].spec}" |jq
```

You can find more information about the current PAM version's component versions by navigating to the website appropriate to your installation type:

- **HA installations:** https://kurl.sh/puppet-application-manager
- **Standalone installations:** https://kurl.sh/puppet-application-manager-standalone

Install PAM

You can install Puppet-supported Puppet Application Manager on a single node or in an HA configuration. Both online and offline install packages are available. You can also install it on an existing Kubernetes cluster.

Refer to the Architecture overview on page 23 for guidance on choosing which Puppet-supported Kubernetes cluster configuration is most appropriate for your needs.

> **Important:** The Puppet-supported Puppet Application Manager cluster brings its own container runtime as part of the kURLinstallation.
>
> - Do not install a container runtime from your operating system (OS) vendor or third-party.
> - Do not install PAM on a node that has previously hosted a container runtime from your OS vendor or third-party.
>
> Installing a different container runtime on a node, even if you installed and removed the packages before you installed PAM, causes failures that persist even after you've uninstalled the runtime.

For information on installing Puppet Application Manager on an existing Kubernetes cluster, see Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 42.

- Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 42
Use these instructions to install Puppet Application Manager and any Puppet applications on an existing Kubernetes cluster.
- PAM HA online installation on page 44
The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.
- PAM HA offline installation on page 48
Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

- PAM standalone online installation on page 51

The Puppet Application Manager (PAM) installation process sets up the application manager (with a simple Kubernetes installation for container orchestration) for you and installs the application on the single-node cluster.

- PAM standalone offline installation on page 54

Use these instructions to install Puppet Application Manager (PAM) in an offline environment where the Puppet Application Manager host server does not have direct access to the internet.

- Automate PAM and Puppet application online installations on page 56

During a fresh online installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

- Automate PAM and Puppet application offline installations on page 58

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

- Uninstall PAM on page 62

Different uninstall procedures are required for Puppet-supported and customer-supported clusters

Related information

Architecture overview on page 23

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

Install Puppet applications using PAM on a customer-supported Kubernetes cluster

Use these instructions to install Puppet Application Manager and any Puppet applications on an existing Kubernetes cluster.

Before you begin

1. If you haven't already done so, install kubectl.
2. Puppet Application Manager is expected to work on any certified Kubernetes distribution that meets the following requirements. We validated and support:
   - Google Kubernetes Engine
   - AWS Elastic Kubernetes Service

   If you use a different distribution, contact Puppet Support for more information on compatibility with PAM.
3. Make sure your Kubernetes cluster meets the minimum requirements:
   - Kubernetes version 1.24-1.26.
   - A default storage class that can be used for relocatable storage.
   - A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
   - We currently test and support Google Kubernetes Engine (GKE) clusters.

   Note: If you're using self-signed certificates on your Ingress controller, you must ensure that your job hardware nodes trust the certificates. Additionally, all nodes that use Continuous Delivery for PE webhooks must trust the certificates, or SSL checking must be disabled on these nodes.

   Important: If you are installing Puppet Comply on Puppet Application Manager, the ingress controller must be configured to allow request payloads of up to 32 MB. Ingress controllers used by Amazon EKS commonly default to a 1 MB maximum — this causes all report submissions to fail.

   The ingress must have a generous limit for total connection time. Setting the connection timeout to `infinity` in conjunction with an idle timeout is recommended.
4. If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via `HTTP_PROXY`/`HTTPS_PROXY`/`NO_PROXY` environment variables.

   Restriction: Using a proxy to connect to external version control systems is currently not supported.

Installation takes several (mostly hands-off) minutes to complete.

1. Install the KOTS (Kubernetes off-the-shelf software) plugin on a workstation that has kubectl access to the cluster. Your kubectl configuration must have sufficient privileges to create cluster-level roles and permissions:

```
curl https://kots.io/install | bash
```

2. If you are performing an offline install, ensure the required images are available in a local registry.
   a) Download the release assets matching the CLI version using the following command:

```
curl -LO https://github.com/replicatedhq/kots/releases/download/v
$(kubectl kots version | head -n1 | cut -d' ' -f3)/kotsadm.tar.gz
```

   b) Extract the images and push them into a private registry. Registry credentials provided in this step must have push access. These credentials are not stored anywhere or reused later.

```
kubectl kots admin-console push-images ./kotsadm.tar.gz
 <private.registry.host>/puppet-application-manager \
--registry-username <rw-username> \
--registry-password <rw-password>
```

   c) Install Puppet Application Manager using images pushed in the previous step. Registry credentials provided in this step only need to have read access, and they are stored in a Kubernetes secret in the current namespace. These credentials are used to pull the images.

```
kubectl kots install puppet-application-manager \
--kotsadm-namespace puppet-application-manager \
--kotsadm-registry <private.registry.host> \
--registry-username <ro-username> \
--registry-password <ro-password>
```

> **Note:** If you are setting up Puppet Application Manager behind a proxy server, add the `--copy-proxy-env` flag to this command to copy the proxy-related environment values from your environment.

   d) You can use similar commands to upload images from the application bundle to your registry to continue to use read-only access when pulling images. Use the same registry namespace (`puppet-application-manager`) to pull application images.

```
kubectl kots admin-console push-images ./<application-release>.airgap
 <private.registry.host>/puppet-application-manager \
--registry-username <rw-username> \
--registry-password <rw-password>
```

3. To perform an online install of Puppet Application Manager on your cluster, run the following commands from a workstation that has kubectl access to the cluster.

```
kubectl kots install puppet-application-manager --namespace <target
 namespace>
```

   This installs Puppet Application Manager on the cluster and sets up a port forward on the ClusterIP.

4. Navigate to `http://localhost:8800` and follow the prompts to be guided through the process of uploading a license for the application, configuring a local registry (for offline installs), checking to make sure your infrastructure meets system requirements, and configuring the application.

> **Note:** If you are performing an offline install, download the application bundle and provide it when prompted.

> **Tip:** Clusters like GKE often restrict ports to 30000-32767. The webhook for Continuous Delivery for PE defaults to port 8000. To update this port to something in the allowed range, when configuring the application, use the following steps:
>
> a. On the Puppet Application Manager **Dashboard** page, under **Config** > **Optional configuration**, select **View options for using a proxy or external load balancer**.
> b. Enter a new value for **Webhook service port**.

5. To configure your installation further, click **Config**. On this tab, you can configure a public hostname, root user, and other settings. These are written as Kubernetes secrets in the deployment manifests. For information on how to configure your application, see the documentation for that application:
   - Configure Continuous Delivery for PE in an online environment
   - Configure Comply in an online environment
6. To use cert-manager, in the **Customize endpoints** section, select **I have cert manager** and in the annotations section, add yours. For example:

```
kubernetes.io/ingress.class: nginx
                    cert-manager.io/cluster-issuer: letsencrypt-prod
```

7. When you are happy with your configuration, click **Save config** to deploy the application.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For general information, go to Install applications via the PAM UI on page 63.

For more information on installing Continuous Delivery for PE online, see Install Continuous Delivery for PE.

For more information on installing Comply online, see Install Comply online.

Related information

Upgrade PAM on a customer-supported online cluster on page 71
Upgrading Puppet Application Manager (PAM) on a customer-supported online Kubernetes cluster can be done with a single command.

Upgrade PAM on a customer-supported offline cluster on page 72
Upgrading Puppet Application Manager (PAM) on a customer-supported offline Kubernetes cluster requires a few simple kubectl commands.

PAM HA online installation

The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.

Before you begin

1. Review the Puppet Application Manager system requirements.
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.

3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

   The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

   If you want to keep Firewalld enabled:

   a. Make sure Firewalld is installed on your system.
   b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
      - ["--zone=external", "--add-masquerade"]
      # SSH port
      - ["--permanent", "--zone=public", "--add-port=22/tcp"]
      # HTTPS port
      - ["--permanent", "--zone=public", "--add-port=443/tcp"]
      # Kubernetes etcd port
      - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
      # Kubernetes API port
      - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
      # Flannel Net port
      - ["--permanent", "--zone=public", "--add-port=8472/udp"]
      # CD4PE Webhook callback port (uncomment line below if needed)
      # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
      # KOTS UI port
      - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
      # CD4PE Local registry port (offline only, uncomment line below if
 needed)
      # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
      # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
      - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
      # Reload firewall rules
      - ["--reload"]
    bypassFirewalldWarning: true
    disableFirewalld: false
    hardFailOnFirewalld: false
    preserveConfig: false
```

4. Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See Resolve IP address range conflicts on page 83 for instructions.

   Note: The minimum size for CIDR blocks used by Puppet Application Manager are:

   - **Standalone** - /24 for pod and service CIDRs
   - **HA** - /23 for pod and service CIDRs
   - Default of /22 is recommended to support future expansion

5. If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via `HTTP_PROXY`/`HTTPS_PROXY`/`NO_PROXY` environment variables.

   Restriction: Using a proxy to connect to external version control systems is currently not supported.

6. Set all nodes used in your HA implementation to the UTC timezone.
7. If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

This installation process results in a Puppet Application Manager instance that is configured for high availability. Installation takes several minutes (mostly hands-off) to complete.

For more context about HA components and structure, refer to the **HA architecture** section of the Architecture overview on page 23.

1. Install and configure a load balancer (or two if you want to segment internal and external traffic - for more information, see Architecture overview on page 23). Round-robin load balancing is sufficient. For an HA cluster, the following is required:

   - A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
   - A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

   Note: Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

2. From the command line of your first primary node, run the installation script:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager | sudo bash
```

Tip: If you're installing Puppet Application Manager behind a proxy server, using `sudo` might cause the installation to fail. Try running the command as root and replace `sudo bash` with `bash`.

Note: An unformatted, unpartitioned storage device is required.

By default this installation automatically uses devices (under `/dev`) matching the pattern `vd[b-z]`. Attach a device to each host. Only devices that match the pattern, and are unformatted, are used.

If necessary, you can override this pattern by providing a patch during installation; append `-s installer-spec-file=patch.yaml` to the installation command.

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  rook:
    blockDeviceFilter: "sd[b-z]"
```

a) When prompted for a load balancer address, enter the address of the DNS entry for your load balancer.
b) The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

```
---
Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
Login with password (will not be shown again): <PASSWORD>
---
```

Note: If you lose this password or wish to change it, see Reset the PAM password on page 83 for instructions.

c) When the installation script is complete, run `bash -l` to reload the shell.

Tip: If the installation script fails, run the following and upload the results to the Puppet Support team:

```
kubectl support-bundle https://kots.io
```

If you're installing as the root user, run the command directly:

```
/usr/local/bin/kubectl-support_bundle https://kots.io
```

3. Add two additional primary nodes to the installation by following the instructions in the install script:

```
To add MASTER nodes to this installation, run the following script on your
 other nodes:
curl -sSL
https://k8s.kurl.sh/puppet-application-manager-unstable/join.sh
| sudo bash -s kubernetes-master-address=...
```

If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the install command.

4. Add the two new nodes to your load balancer.

5. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

   The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 63.

For more information on installing Continuous Delivery for PE online, see Install Continuous Delivery for PE.

For more information on installing Comply online, see Install Comply online.

Related information

Reset the PAM password on page 83
As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

PAM system requirements on page 27
You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Resolve IP address range conflicts on page 83
When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

Architecture overview on page 23
Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

PAM HA offline installation

Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

Before you begin

1. Review the Puppet Application Manager system requirements.
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.
3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

   The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

   If you want to keep Firewalld enabled:

   a. Make sure Firewalld is installed on your system.
   b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
```

```
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
      - ["--zone=external", "--add-masquerade"]
      # SSH port
      - ["--permanent", "--zone=public", "--add-port=22/tcp"]
      # HTTPS port
      - ["--permanent", "--zone=public", "--add-port=443/tcp"]
      # Kubernetes etcd port
      - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
      # Kubernetes API port
      - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
      # Flannel Net port
      - ["--permanent", "--zone=public", "--add-port=8472/udp"]
      # CD4PE Webhook callback port (uncomment line below if needed)
      # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
      # KOTS UI port
      - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
      # CD4PE Local registry port (offline only, uncomment line below if
 needed)
      # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
      # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
      - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
      # Reload firewall rules
      - ["--reload"]
    bypassFirewalldWarning: true
    disableFirewalld: false
    hardFailOnFirewalld: false
    preserveConfig: false
```

4. Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See Resolve IP address range conflicts on page 83 for instructions.

   Note: The minimum size for CIDR blocks used by Puppet Application Manager are:

   - **Standalone** - /24 for pod and service CIDRs
   - **HA** - /23 for pod and service CIDRs
   - Default of /22 is recommended to support future expansion

5. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
6. Set all nodes used in your HA implementation to the UTC timezone.
7. If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.
8. If you're restoring a backup from a previous cluster, make sure you include the `kurl-registry-ip=<YOUR_IP_ADDRESS>` installation option. For more information, see Migrating PAM data to a new system on page 75.

This installation process results in a basic Puppet Application Manager instance that is configured for optional high availability. Installation takes several minutes (mostly hands-off) to complete.

For more context about HA components and structure, refer to the **HA architecture** section of the Architecture overview on page 23.

1. Install and configure a load balancer (or two if you want to segment internal and external traffic - for more information, see Architecture overview on page 23). Round-robin load balancing is sufficient. For an HA cluster, the following is required:
   - A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
   - A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

   > **Note:** Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

2. From a workstation with internet access, download the cluster installation bundle (note that this bundle is ~4GB):

   ```
   https://k8s.kurl.sh/bundle/puppet-application-manager.tar.gz
   ```

3. Copy the installation bundle to your primary and secondary nodes and unpack it:

   ```
   tar xzf puppet-application-manager.tar.gz
   ```

4. Run the installation command:

   ```
   cat install.sh | sudo bash -s airgap
   ```

   > **Note:** An unformatted, unpartitioned storage device is required.
   >
   > By default this installation automatically uses devices (under `/dev`) matching the pattern `vd[b-z]`. Attach a device to each host. Only devices that match the pattern, and are unformatted, are used.
   >
   > If necessary, you can override this pattern by providing a patch during installation; append `-s installer-spec-file=patch.yaml` to the installation command.
   >
   > ```
   > apiVersion: cluster.kurl.sh/v1beta1
   > kind: Installer
   > metadata:
   >  name: patch
   > spec:
   >  rook:
   >  blockDeviceFilter: "sd[b-z]"
   > ```

   a) When prompted for a load balancer address, enter the address of the DNS entry for your load balancer.
   b) The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

   ```
   ---
   Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
   Login with password (will not be shown again): <PASSWORD>
   ---
   ```

   > **Note:** If you lose this password or wish to change it, see Reset the PAM password on page 83 for instructions.

5. Add two additional primary nodes to your offline installation using the instructions provided in the install script:

```
To add MASTER nodes to this installation, copy and unpack this bundle on
 your other nodes, and run the following:
cat ./join.sh | sudo bash -s airgap
kubernetes-master-address=...
```

6. Add the two new nodes to your load balancer.
7. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

   The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for installing your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 63.

For more information on installing Continuous Delivery for PE offline, see Install Continuous Delivery for PE in an offline environment.

For more information on installing Comply offline, see Install Comply offline.

Related information

Reset the PAM password on page 83
As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

PAM system requirements on page 27
You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Resolve IP address range conflicts on page 83
When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

Architecture overview on page 23
Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

PAM standalone online installation

The Puppet Application Manager (PAM) installation process sets up the application manager (with a simple Kubernetes installation for container orchestration) for you and installs the application on the single-node cluster.

Before you begin

1. Review the Puppet Application Manager system requirements.
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.

3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

   The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

   If you want to keep Firewalld enabled:

   a. Make sure Firewalld is installed on your system.
   b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
      - ["--zone=external", "--add-masquerade"]
      # SSH port
      - ["--permanent", "--zone=public", "--add-port=22/tcp"]
      # HTTPS port
      - ["--permanent", "--zone=public", "--add-port=443/tcp"]
      # Kubernetes etcd port
      - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
      # Kubernetes API port
      - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
      # Flannel Net port
      - ["--permanent", "--zone=public", "--add-port=8472/udp"]
      # CD4PE Webhook callback port (uncomment line below if needed)
      # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
      # KOTS UI port
      - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
      # CD4PE Local registry port (offline only, uncomment line below if
 needed)
      # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
      # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
      - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
      # Reload firewall rules
      - ["--reload"]
    bypassFirewalldWarning: true
    disableFirewalld: false
    hardFailOnFirewalld: false
    preserveConfig: false
```

4. Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See Resolve IP address range conflicts on page 83 for instructions.
5. If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

This installation process results in a basic Puppet Application Manager instance. Installation takes several (mostly hands-off) minutes to complete.

1. From the command line of your node, run the installation script:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager-standalone | sudo
 bash
```

> **Tip:** If you're installing Puppet Application Manager behind a proxy server, using `sudo` might cause the installation to fail. Try running the command as root (use command `sudo su -`) and replace `sudo bash` with `bash`.

   a) When the installation script prints the Puppet Application Manager address and password, make a careful note of these credentials:

```
---
Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
Login with password (will not be shown again): <PASSWORD>
---
```

   > **Note:** If you lose this password or wish to change it, see Reset the PAM password on page 83 for instructions.

   b) When the installation script is complete, run `bash -l` to reload the shell.

> **Tip:** If the installation script fails, run the following and upload the results to the Puppet Support team:
>
> ```
> kubectl support-bundle https://kots.io
> ```
>
> If you're installing as the root user, run the command directly:
>
> ```
> /usr/local/bin/kubectl-support_bundle https://kots.io
> ```

2. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

   The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 63.

For more information on installing Continuous Delivery for PE online, see Install Continuous Delivery for PE.

For more information on installing Comply online, see Install Comply online.

Related information

Reset the PAM password on page 83
As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

PAM system requirements on page 27
You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Resolve IP address range conflicts on page 83
When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

Architecture overview on page 23
Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

PAM standalone offline installation

Use these instructions to install Puppet Application Manager (PAM) in an offline environment where the Puppet Application Manager host server does not have direct access to the internet.

Before you begin

1. Review the Puppet Application Manager system requirements.
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.
3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

   The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

   If you want to keep Firewalld enabled:

   a. Make sure Firewalld is installed on your system.
   b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
      - ["--zone=external", "--add-masquerade"]
      # SSH port
      - ["--permanent", "--zone=public", "--add-port=22/tcp"]
      # HTTPS port
      - ["--permanent", "--zone=public", "--add-port=443/tcp"]
      # Kubernetes etcd port
      - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
      # Kubernetes API port
      - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
      # Flannel Net port
      - ["--permanent", "--zone=public", "--add-port=8472/udp"]
      # CD4PE Webhook callback port (uncomment line below if needed)
      # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
      # KOTS UI port
      - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
      # CD4PE Local registry port (offline only, uncomment line below if
 needed)
      # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
      # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
```

```
      - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
      # Reload firewall rules
      - ["--reload"]
    bypassFirewalldWarning: true
    disableFirewalld: false
    hardFailOnFirewalld: false
    preserveConfig: false
```

4. Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See Resolve IP address range conflicts on page 83 for instructions.
5. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
6. If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.
7. If you're restoring a backup from a previous cluster, make sure you include the `kurl-registry-ip=<YOUR_IP_ADDRESS>` installation option. For more information, see Migrating PAM data to a new system on page 75.

This installation process results in a basic Puppet Application Manager instance. Installation takes several (mostly hands-off) minutes to complete.

1. From a workstation with internet access, download the cluster installation bundle (note that this bundle is ~4GB):

   ```
   https://k8s.kurl.sh/bundle/puppet-application-manager-standalone.tar.gz
   ```

2. Copy the installation bundle to the host node and unpack it:

   ```
   tar xzf puppet-application-manager-standalone.tar.gz
   ```

3. Run the installation command:

   ```
   cat install.sh | sudo bash -s airgap
   ```

   a) The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

   ```
   ---
   Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
   Login with password (will not be shown again): <PASSWORD>
   ---
   ```

   Note: If you lose this password or wish to change it, see Reset the PAM password on page 83 for instructions.

4. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

   The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 63.

For more information on installing Continuous Delivery for PE offline, see Install Continuous Delivery for PE in an offline environment.

For more information on installing Comply offline, see Install Comply offline.

Related information

Reset the PAM password on page 83

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

PAM system requirements on page 27
You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Resolve IP address range conflicts on page 83
When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

Architecture overview on page 23
Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Automate PAM and Puppet application online installations

During a fresh online installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

> **Before you begin**
>
> Ensure that your system meets the PAM system requirements on page 27.

1. Install Puppet Application Manager. For detailed instructions, see PAM HA online installation on page 44.

2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
    annotations:
      value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
    hostname:
      value: "<HOSTNAME>"
    root_password:
      value: "<ROOT ACCOUNT PASSWORD>"
```

Tip: View the keyword names for all settings by clicking **View files** > **upstream** > **config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace `<HOSTNAME>` with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace `<ROOT ACCOUNT PASSWORD>` your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- **Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at https://puppet.com/legal.

3. Write your license file and the configuration values generated in step 1 to the following locations:
   - Write your license file to `./replicated_license.yaml`
   - Write your configuration values to `./replicated_config.yaml`
4. Add the Puppet application definition to Puppet Application Manager with the license file and configuration values, passing in the Puppet Application Manager password you set in step 4:

```
kubectl kots install <APPLICATION NAME> --namespace default --shared-
password <YOUR CHOSEN PASSWORD> --port-forward=false \
  --license-file ./replicated_license.yaml --config-values ./
replicated_config.yaml
```

Note: If you want to install a specific version of the application, include the `--app-version-label=<VERSION>` flag in the install command.

5. Wait five minutes to allow the software time to process the change.

6. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

   Your configuration values are applied, and if preflight checks have passed, the application is deployed and in the process of starting up.

   The application's status on the **Application** tab is shown as **Missing** for several minutes while deployment is underway. To monitor the deployment's progress, run `kubectl get pods --watch`.

   When the deployment is complete, the application status changes to **Ready**.
7. Update your DNS or `/etc/hosts` file to include the hostname you chose during configuration.
8. Installation is now complete! Navigate to `https://<HOSTNAME>` and sign into Puppet application.

Related information

PAM HA online installation on page 44
The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.

Upgrade an automated online application installation on page 64
If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

Automate PAM and Puppet application offline installations

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Before you begin

Ensure that your system meets the PAM system requirements on page 27.

Related information

PAM HA offline installation on page 48
Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

Upgrade an automated offline application installation on page 65
If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

Automate PAM and Puppet application offline installations on Puppet-supported clusters

1. Install Puppet Application Manager. For detailed instructions, see PAM HA offline installation on page 48.

2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
    annotations:
      value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
    hostname:
      value: "<HOSTNAME>"
    root_password:
      value: "<ROOT ACCOUNT PASSWORD>"
```

Tip: View the keyword names for all settings by clicking **View files** > **upstream** > **config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace `<HOSTNAME>` with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace `<ROOT ACCOUNT PASSWORD>` your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- **Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at https://puppet.com/legal.

3. Write your license file and the configuration values generated in the previous step to the following locations:
   - Write your license file to `./replicated_license.yaml`
   - Write your configuration values to `./replicated_config.yaml`
4. Download the application bundle:

```
curl -L <APPLICATION BUNDLE URL> -o <APPLICATION BUNDLE FILE>
```

5. Copy the application bundle to your primary and secondary nodes and unpack it:

```
tar xzf ./<APPLICATION BUNDLE FILE>
```

6. Run the application install command on your primary node. Replace the `<YOUR CHOSEN PASSWORD>` , `<APPLICATION NAME>`, `<APPLICATION BUNDLE FILE>` values in the example below with your own values:

```
KOTS_PASSWORD=<YOUR CHOSEN PASSWORD>
kubectl kots install <APPLICATION NAME> --namespace default --shared-
password $KOTS_PASSWORD --license-file ./license.yaml --config-
values ./config.yaml --airgap-bundle ./<APPLICATION BUNDLE FILE> --port-
forward=false
# wait several minutes for the application to deploy; if it doesn't show
 up, preflights or another error might have occurred
```

Note: If you want to install a specific version of the application, include the `--app-version-label=<VERSION>` flag in the install command.

Automate PAM and Puppet application offline installations on customer-supported clusters

Before you begin

1. If you haven't already done so, install kubectl.
2. Puppet Application Manager is expected to work on any certified Kubernetes distribution that meets the following requirements. We validated and support:
   - Google Kubernetes Engine
   - AWS Elastic Kubernetes Service

   If you use a different distribution, contact Puppet Support for more information on compatibility with PAM.
3. Make sure your Kubernetes cluster meets the minimum requirements:
   - Kubernetes version 1.24-1.26.
   - A default storage class that can be used for relocatable storage.
   - A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
   - We currently test and support Google Kubernetes Engine (GKE) clusters.

   Note: If you're using self-signed certificates on your Ingress controller, you must ensure that your job hardware nodes trust the certificates. Additionally, all nodes that use Continuous Delivery for PE webhooks must trust the certificates, or SSL checking must be disabled on these nodes.

   Important: If you are installing Puppet Comply on Puppet Application Manager, the ingress controller must be configured to allow request payloads of up to 32 MB. Ingress controllers used by Amazon EKS commonly default to a 1 MB maximum — this causes all report submissions to fail.

   The ingress must have a generous limit for total connection time. Setting the connection timeout to `infinity` in conjunction with an idle timeout is recommended.
4. If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via `HTTP_PROXY`/`HTTPS_PROXY`/`NO_PROXY` environment variables.

   Restriction: Using a proxy to connect to external version control systems is currently not supported.

1. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
    annotations:
      value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
    hostname:
      value: "<HOSTNAME>"
    root_password:
      value: "<ROOT ACCOUNT PASSWORD>"
```

Tip: View the keyword names for all settings by clicking **View files** > **upstream** > **config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace `<HOSTNAME>` with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace `<ROOT ACCOUNT PASSWORD>` your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- **Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at https://puppet.com/legal.

2. Write your license file and the configuration values generated in the previous step to the following locations:
   - Write your license file to `./replicated_license.yaml`
   - Write your configuration values to `./replicated_config.yaml`
3. Download the application bundle:

```
curl -L <APPLICATION BUNDLE URL> -o <APPLICATION BUNDLE FILE>
```

4. Create and run the following script, supplying values specific to your installation for the variables:

```
#!/bin/bash
REGISTRY=<YOUR_CONTAINER_REGISTRY>
APP_K8S_NAMESPACE=<DESIRED_NAMESPACE_IN_TARGET_CLUSTER>
APP_BUNDLE=<PATH_TO_AIRGAP_BUNDLE_FROM_STEP_3>
PAM_PASSWORD=<DESIRED_PAM_CONSOLE_PASSWORD>
LICENSE_FILE=<PATH_TO_LICENSE_FILE_FROM_STEP_1>
CONFIG_FILE=<PATH_TO_CONFIG_FILE_FROM_STEP_2>

curl https://kots.io/install | bash
curl -LO https://github.com/replicatedhq/kots/releases/download/v$(kubectl
 kots version | head -n1 | cut -d' ' -f3)/kotsadm.tar.gz

kubectl kots admin-console push-images ./kotsadm.tar.gz ${REGISTRY}
kubectl kots admin-console push-images ${APP_BUNDLE} ${REGISTRY}
kubectl kots install puppet-application-manager --namespace
 ${APP_K8S_NAMESPACE} --shared-password ${PAM_PASSWORD} --license-
file ${LICENSE_FILE} --config-values ${CONFIG_FILE} --airgap-bundle
 ${APP_BUNDLE} --disable-image-push --kotsadm-registry ${REGISTRY} --port-
forward=false --skip-preflights
```

Tip: If the script fails, it might be because:

- The `push-images` commands require that the local machine where the script is running has `push` access to the registry.
- The `install` command requires `read` access to the registry from the target cluster.
- Offline HA installs of GKE can't run preflights; therefore `--skip-preflights` must be included.

Uninstall PAM

Different uninstall procedures are required for Puppet-supported and customer-supported clusters

At this time it's not possible to cleanly uninstall PAM from Puppet-supported clusters.

If you need to start with a fresh PAM install, you'll need to provision a new host.

Uninstall PAM on customer-supported clusters

To uninstall Puppet Application Manager from customer-supported clusters, use:

```
kubectl delete namespace <pam-namespace>
kubectl delete clusterrolebinding kotsadm-rolebinding
kubectl delete clusterrole kotsadm-role
```

Related information

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Working with Puppet applications

You can install and upgrade Puppet applications using the Puppet Application Manager UI.

Install applications via the PAM UI

The process of adding an application once you've installed Puppet Application Manager is simple.

Important: Ensure you are using the following Puppet application versions if you want to add more than one Puppet application via the Puppet Application Manager UI:

Application	Version
Continuous Delivery for Puppet Enterprise	4.6.0 or later
Comply	1.0.4 or later

For information on installing Puppet applications via the command line, see Automate PAM and Puppet application online installations on page 56 and Automate PAM and Puppet application offline installations on page 58.

To install a Puppet application using the Puppet Application Manager UI:

1. Log into the Puppet Application Manager UI, and click **Add a new application**.
   - If you have not added a Puppet application before you are prompted to upload a license.
   - If you have already added a Puppet application, click **Add a new application**.
2. Upload your `replicated_license.yaml` file when requested.

Note: Once the license file is installed, if offline installations are enabled, you are presented with an option to proceed with an offline setup.

Add the following information to install an offline application:

- **Hostname** - the hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.

  Important: The hostname must be unique for each application you install.
- **Username**/**Password** - The username and password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Registry namespace** - the registry namespace for the application, e.g. *CD4PE* or *Comply*.
- **Airgap bundle** - upload the relevant application bundle tarball. Click **Continue**.

3. Add any additional required information that is presented on the **Config** page. Configure any other settings on the page relevant to your installation, such as external databases, customized endpoints, a load balancer, or TLS certificates. Click **Save Config** when you are done.

   Saving your new configuration settings prompts the creation of a new application version.
4. Click **Go to new version**, which redirects you to the **Version history** tab. The newly created version is shown in the **All versions** section of the page.
5. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while your system is checked to make sure your cluster meets minimum system requirements. When the preflight check is complete:
   - If the status is **Checks Failed**, click **View preflights**. Correct the issues and click **Re-run**. Repeat this step as needed.

     > **Important:** Do not move on until all preflight checks pass.
   - If the status is **Ready to Deploy**, move on to the next step.
6. Once the version is ready to deploy, click **Deploy**. On the **Application** tab, monitor the application for readiness.

   The application's status is shown as **Missing** for several minutes while deployment is underway. To monitor the deployment's progress, run `kubectl get pods --watch`.

   When the deployment is complete, the application status changes to **Ready**.
7. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into your application.

Update a license for online installations

If you have performed online installation of an application, you can use the Puppet Application Manager UI to update your license.

To update the license for an online application:

1. Log in to Puppet Application Manager, click the **License** tab, and then **Sync License**.
2. On the **Version history** tab, click **Deploy**.

   Puppet Application Manager adds “License Change” as the deployment cause on the **Version history** tab.

Update a license for offline installations

If you have performed offline installation of an application, you can use the Puppet Application Manager UI to update your license.

To update the license for an offline application:

1. Ask your Puppet sales representative to email you an updated license file.
2. Log in to Puppet Application Manager, click the **License** tab.
3. Drag and drop or upload the updated license file provided by your Puppet sales representative.
4. On the **Version history** tab, click **Deploy**.

   Puppet Application Manager adds “License Change” as the deployment cause on the **Version history** tab.

Upgrade an automated online application installation

If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

> **Important:** Ensure that you are following an approved upgrade path for the application you want to upgrade. For more information, check the relevant application documentation.

1. From the command line of your primary (control plane) node, get the *application slug* for the application you want to upgrade:

   ```
   kubectl kots --namespace <NAMESPACE> get apps
   ```

   Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually `default`).
2. Run the upgrade script:

   ```
   kubectl kots upstream upgrade <APPLICATION SLUG> --namespace <NAMESPACE>
    --deploy
   ```

   Replace `<APPLICATION SLUG>` with the relevant application slug for the application you want to upgrade.

   Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually `default`).
3. Wait five minutes to allow the software time to process the change.
4. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

   If preflight checks have passed, the upgraded application is deployed and in the process of starting up. To monitor the deployment's progress, run:

   ```
   kubectl get pods --watch
   ```

Related information

Automate PAM and Puppet application offline installations on page 58
During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Upgrade an automated offline application installation

If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

> **Important:** Ensure that you are following an approved upgrade path for the application you want to upgrade. For more information, check the relevant application documentation.

1. Download the application bundle you want to upgrade to. Copy to your primary node.
2. From the command line of your primary (control plane) node, get the *application slug* for the application you want to upgrade:

   ```
   kubectl kots --namespace <NAMESPACE> get apps
   ```

   Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually `default`).
3. Run the upgrade script:

   ```
   kubectl kots upstream upgrade <APPLICATION SLUG> --airgap-bundle ./
   <APPLICATION BUNDLE FILE> --kotsadm-namespace <REGISTRY NAMESPACE> --
   namespace <NAMESPACE> --deploy
   ```

   - Replace `<APPLICATION SLUG>` with the relevant application slug for the application you want to upgrade.
   - Replace `<APPLICATION BUNDLE FILE>` with the name of the application bundle file.
   - Replace `<REGISTRY NAMESPACE>` with your Registry namespace where images are uploaded.
   - Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually `default`).
4. Wait five minutes to allow the software time to process the change.

5. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

   If preflight checks have passed, the upgraded application is deployed and in the process of starting up. To monitor the deployment's progress, run:

   ```
   kubectl get pods --watch
   ```

Related information

Automate PAM and Puppet application offline installations on page 58
During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Maintenance and tuning

Follow these guidelines when you're tuning or performing maintenance on a node running Puppet Application Manager (PAM).

How to look up your Puppet Application Manager architecture

If you're running PAM on a Puppet-supported cluster, you can use the following command to determine your PAM architecture version:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o
 jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which architecture you used when installing, the command returns one of these values:

- **HA architecture**: `puppet-application-manager`
- **Standalone architecture**: `puppet-application-manager-standalone`
- **Legacy architecture**: Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`

Rebooting PAM nodes

Where possible, avoid rebooting or shutting down a PAM node. Shutting down an HA PAM node incorrectly could result in storage volume corruption and the loss of data.

For tasks such as package updates or security patches, where you must perform a reboot or shut down, follow the procedure below to gracefully shut down the node and ensure that it is drained correctly.

To reboot a node:

1. Shut down services using Ceph-backed storage:

   ```
   /opt/ekco/shutdown.sh
   ```

2. If you're using a high availability (HA) cluster, drain the node:

   ```
   kubectl drain <NODE NAME> --ignore-daemonsets --delete-local-data
   ```

3. Reboot the node.

Load balancer health checks

To set up health checks for the load balancer that your Puppet Application Manager (PAM) applications are running behind, set up rules for these applications and services.

Application/service	URL/port	Notes
Puppet application. For example, Continuous Delivery for Puppet Enterprise or Puppet Comply	`https://<CDPE HOSTNAME>:443/status`	Although Puppet applications might expose other ports (Continuous Delivery for PE exposes ports 443, 80, and 8000), 443 is the HTTPS endpoint, and is the best port to use for health checks.
Puppet Application Manager (PAM)	`https://<KUBERNETES PRIMARY IP>:8800/healthz`	
External load balancer endpoint	Port `6443` or `https://<KUBERNETES PRIMARY IP>:6443/livez`	For information on setting up a TCP probe on an external load balancer endpoint, consult the kURL load balancer documentation.
Local container registry (for offline installations)	`https://<KUBERNETES PRIMARY IP>:9001`	

Load balancing

The following load balancer requirements are needed for a HA install:

- A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
- A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

> **Note:** Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

Important: If you are using application load balancing, be aware that Ingress items use Server Name Indication (SNI) to route requests, which may require additional configuration with your load balancer. If your load balancer does not support SNI for health checks, enable **Enable load balancer HTTP health check** in the Puppet Application Manager UI **Config** page .

Upgrading PAM on a Puppet-supported cluster

Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

There are four possible upgrade types for Puppet Application Manager installations:

- **Online** - For standalone or HA installations with a connection to the internet.
- **Offline** - For air-gapped standalone or HA installations without a connection to the internet.
- **Online legacy** - For standalone or HA installations created prior to April 2021 with a connection to the internet.
- **Offline legacy** - For air-gapped standalone or HA installations created prior to April 2021 without a connection to the internet.

Restriction: You cannot use the upgrade process to move from a legacy deployment to a non-legacy deployment, or from standalone to HA, or vice versa. If you wish to change architecture types, see Migrating PAM data to a new system on page 75.

How to look up your Puppet Application Manager architecture

If you're running PAM on a Puppet-supported cluster, you can use the following command to determine your PAM architecture version:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o
 jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which architecture you used when installing, the command returns one of these values:

- **HA architecture**: `puppet-application-manager`
- **Standalone architecture**: `puppet-application-manager-standalone`
- **Legacy architecture**: Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`

Upgrade PAM online

Upgrade Puppet Application Manager (PAM) to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 73.

If you are upgrading from a version of PAM that used Weave (versions 1.100.3 and earlier) to a version of PAM that uses Flannel (versions 1.102.2 and later), pod-to-pod networking now depends on UDP port 8472 being open instead of ports 6783 and 6784.

Note: Starting with Puppet Application Manager 1.97.0, the `force-reapply-addons` flag is deprecated and generates a warning on use. If you are upgrading to a version prior to 1.97.0, you need to add the `force-reapply-addons` flag to the `bash` command using the `-s` flag.

1. On your first primary node, rerun the installation script, passing in any arguments you included when installing for the first time:

   For standalone deployments, use:

   ```
   curl -sSL https://k8s.kurl.sh/puppet-application-manager-standalone | sudo
    bash
   ```

   For HA deployments, use:

   ```
   curl -sSL https://k8s.kurl.sh/puppet-application-manager | sudo bash
   ```

2. If a new version of Kubernetes is available, the installer notes upgrade scripts to run on other nodes in an HA cluster.

   The installer also pauses before draining nodes as part of the Kubernetes upgrade. The node draining process can take several minutes to complete, during which time application workloads are stopped or migrated to other systems. This migration may cause several minutes of downtime while databases are rescheduled.

Related information

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Upgrade PAM offline

Users operating in environments without direct access to the internet must use the links below to upgrade to the latest version of Puppet Application Manager (PAM).

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 73.

If you are upgrading from a version of PAM that used Weave (versions 1.100.3 and earlier) to a version of PAM that uses Flannel (versions 1.102.2 and later), pod-to-pod networking now depends on UDP port 8472 being open instead of ports 6783 and 6784.

Note: Starting with Puppet Application Manager 1.97.0, the `force-reapply-addons` flag is deprecated and generates a warning on use. If you are upgrading to a version prior to 1.97.0, you need to add the `force-reapply-addons` flag in **Step 3** to the `bash` command after `-s airgap`.

To upgrade Puppet Application Manager:

1. From a workstation with internet access, download the latest version of the installation bundle that is relevant for your installation type:

   For standalone installations, enter the following command (note that this bundle is ~4GB):

   ```
   curl -LO https://k8s.kurl.sh/bundle/puppet-application-manager-standalone.tar.gz
   ```

   For HA installations, enter the following command (note that this bundle is ~4GB):

   ```
   curl -LO https://k8s.kurl.sh/bundle/puppet-application-manager.tar.gz
   ```

2. Copy the installation bundle to your primary and secondary nodes and unpack it:

   For standalone installations, use:

   ```
   tar xzf puppet-application-manager-standalone.tar.gz
   ```

   For HA installations, use:

   ```
   tar xzf puppet-application-manager.tar.gz
   ```

3. Manually load the images from the installation bundle:

   ```
   cat tasks.sh | bash -s load-images
   ```

4. On your primary node, rerun the installation script, passing in any arguments you included when installing for the first time:

   ```
   cat install.sh | sudo bash -s airgap
   ```

   Note: This script issues a prompt to run the `task.sh` and `upgrade.sh` scripts on your secondary nodes. Use the versions of these scripts from the downloaded bundle in step 2.

5. If a new version of Kubernetes is available, the installer systems provide upgrade scripts to run on other nodes in an HA cluster. The installer also pauses before draining nodes as part of the Kubernetes upgrade. Node draining is performed as part of a Kubernetes upgrade.

   The node draining process can take several minutes to complete, during which time application workloads are stopped or migrated to other systems. This migration may cause several minutes of downtime while databases are rescheduled.

When the deployment is complete, sign into Puppet Application Manager- `http://<PUPPET APPLICATION MANAGER ADDRESS>:8800` - and verify that the new version number is displayed in the bottom left corner of the web UI.

PAM legacy upgrades

The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM).

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 73.

Legacy architecture is no longer supported: The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. The legacy architecture reached the end of its support lifecycle on **30 June 2022**, and Puppet no longer updates legacy architecture components. For information on migrating data from a legacy architecture to a standalone or HA architecture, go to our Support Knowledge Base instructions:

- Migrate to a supported PAM architecture for Continuous Delivery for PE
- Migrate to a supported PAM architecture for Comply

Restriction: It is not possible to upgrade from an online legacy install to a new offline install configuration. Similarly, upgrades from an offline legacy configuration to a new online install are not supported.

To upgrade a legacy version of Puppet Application Manager on nodes with internet access:

1. On your node (or control plane node if you have a HA deployment), rerun the installation script, passing in any arguments you included when installing for the first time:
   - For standalone installs:

     ```
     curl -sSL https://k8s.kurl.sh/puppet-application-manager-legacy | sudo
      bash -s force-reapply-addons
     ```

   - For HA installs:

     ```
     curl -sSL https://k8s.kurl.sh/puppet-application-manager-legacy | sudo
      bash -s ha force-reapply-addons
     ```

2. If a new version of Kubernetes is available, the systems provide upgrade scripts to run on each node in your cluster.

   Node draining is performed as part of a Kubernetes upgrade. The node draining process can take several minutes to complete.

   Note: During the Kubernetes upgrade process, nodes are not able to properly route network connections. If you have a HA deployment, make sure you have load balancers or a multi-node fail-over process in place, or schedule downtime before upgrading.

PAM offline legacy upgrades

The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM) on offline nodes.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 73.

Legacy architecture is no longer supported: The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. The legacy architecture reached the end of its support lifecycle on **30 June 2022**, and Puppet no longer updates legacy architecture components. For information on migrating data from a legacy architecture to a standalone or HA architecture, go to our Support Knowledge Base instructions:

- Migrate to a supported PAM architecture for Continuous Delivery for PE
- Migrate to a supported PAM architecture for Comply

Restriction: It is not possible to upgrade from an online legacy install to a new offline install configuration. Similarly, upgrades from an offline legacy configuration to a new online install are not supported.

To upgrade Puppet Application Manager on nodes without a connection to the internet:

1. From a workstation with internet access, download the latest version of the cluster installation bundle (note that this bundle is ~4GB):

   ```
   https://k8s.kurl.sh/bundle/puppet-application-manager-legacy.tar.gz
   ```

2. Copy the installation bundle to your primary and secondary Puppet Application Manager nodes and unpack it:

   ```
   tar xzf puppet-application-manager-legacy.tar.gz
   ```

3. Rerun the installation script. Don't forget to pass in any additional arguments you included when installing for the first time you installed the product:

   For standalone installs use:

   ```
   cat install.sh | sudo bash -s airgap force-reapply-addons
   ```

   For HA installs use:

   ```
   cat install.sh | sudo bash -s airgap ha force-reapply-addons
   ```

   Note: During the upgrade process, follow any prompts to run commands on your other cluster nodes.

When the deployment is complete, sign into Puppet Application Manager and verify that the new version number is displayed in the bottom center of the web UI.

Upgrading PAM on a customer-supported cluster

Upgrade Puppet Application Manager (PAM) on your own Kubernetes cluster to take advantage of new features and bug fixes.

There are two possible upgrade types for customer-supported Puppet Application Manager deployments:

- **Online** - For installations with a connection to the internet.
- **Offline** - For air-gapped installations without a connection to the internet.

Upgrade PAM on a customer-supported online cluster

Upgrading Puppet Application Manager (PAM) on a customer-supported online Kubernetes cluster can be done with a single command.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 73.

To upgrade Puppet Application Manager on a customer-supported online cluster:

1. Upgrade kubectl KOTS:

```
curl https://kots.io/install | bash
```

2. Issue the following KOTS command:

```
kubectl kots admin-console upgrade --namespace <target namespace>
```

Tip: Run the `kubectl kots admin-console upgrade -h` command for more usage information.

Upgrade PAM on a customer-supported offline cluster

Upgrading Puppet Application Manager (PAM) on a customer-supported offline Kubernetes cluster requires a few simple kubectl commands.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 73.

To upgrade Puppet Application Manager on a customer-supported offline cluster, perform the following steps from a workstation that has kubectl access to the cluster:

1. Upgrade kubectl KOTS:

```
curl https://kots.io/install | bash
```

2. Ensure the required images are available in your local registry. Download the release assets matching the CLI version using the following command:

```
curl -LO https://github.com/replicatedhq/kots/releases/download/v$(kubectl
 kots version | head -n1 | cut -d' ' -f3)/kotsadm.tar.gz
```

3. Extract the images and push them to your private registry. Registry credentials provided in this step must have push access. These credentials are not stored anywhere or reused later.

```
kubectl kots admin-console push-images ./kotsadm.tar.gz
 <private.registry.host>/puppet-application-manager \
--registry-username <rw-username> \
--registry-password <rw-password>
```

4. After you push the images to your private registry, execute the upgrade command with registry read-only credentials:

```
kubectl kots upgrade puppet-application-manager \
--kotsadm-namespace puppet-application-manager \
--kotsadm-registry <private.registry.host> \
--registry-username <ro-username> \
--registry-password <ro-password> \
--namespace <target namespace>
```

Backing up PAM using snapshots

Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

Related information

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Full and partial snapshots

There are two options available when you're creating a snapshot for your Puppet Application Manager (PAM) deployment, full snapshots (also known as instance snapshots) and partial (or application) snapshots. For full disaster recovery, make sure you've configured and scheduled regular full snapshots stored on a remote storage solution such as an S3 bucket or NFS share.

Full snapshots offer a comprehensive backup of your PAM deployment, because they include the core PAM application together with the Puppet applications you've installed in yourPAM deployment. You can use a full snapshot to restore your PAM deployment and all of your installed Puppet applications to a previous backup. For example, you could use a full snapshot to revert an undesired configuration change or a failed upgrade, or to migrate your PAM deployment to another Puppet-supported cluster.

Partial snapshots are available from the PAM console, but are limited in their usefulness. To restore from a partial snapshot, you must already have an installed and functioning version of PAM. A functioning PAM installation is needed because the option to restore a partial snapshot can only be accessed from the **Snapshots** section of the PAM admin console.

Partial snapshots only back up the Puppet application you specified when you configured the snapshot, for example, Continuous Delivery for Puppet Enterprise, or Puppet Comply. They do not back up the underlying PAM deployment. Partial snapshots are sometimes useful if you want to roll back to a previous version of a specific Puppet application that you've installed on your PAM deployment, but are far less versatile than full snapshots. To make sure that you have all disaster recovery options available to you, use a full snapshot wherever possible.

Configure snapshots

Before using snapshots, select a storage location, set a snapshot retention period, and indicate whether snapshots are created manually or on a set schedule.

> **Important:** Disaster recovery requires that the store backend used for backups is accessible from the new cluster. When setting up snapshots in an *offline* cluster, make sure to record the registry service IP address with the following command:
>
> ```
> kubectl -n kurl get svc registry -o jsonpath='{.spec.clusterIP}'
> ```
>
> Be sure to record the value returned by this command as it is required when creating a new cluster to restore to as part of Disaster recovery with PAM on page 81.

1. In the upper navigation bar of the Puppet Application Manager UI, click **Snapshots** > **Settings & Schedule**.
2. The snapshots feature uses https://velero.io, an open source backup and restore tool. Click **Check for Velero** to determine whether Velero is present on your cluster, and to install it if needed.

3. Select a destination for your snapshot storage and provide the required configuration information. You can choose to set up snapshot storage in the PAM UI or on the command line. Supported destinations are listed below. We recommend using an external service or NFS, depending on what is available to you:

   - Internal storage (default)
   - Amazon S3
   - Azure Blob Storage
   - Google Cloud Storage
   - Other S3-compatible storage
   - Network file system (NFS)
   - Host path

   Amazon S3 storage

   If using the PAM UI, provide the following information:

Field	Description
Bucket	The name of the AWS bucket where snapshots are stored.
Region	The AWS region the bucket is available in.
Path	**Optional.** The path within the bucket where all snapshots are stored.
Use IAM instance role?	If selected, an IAM instance role is used instead of an access key ID and secret.
Access key ID	**Required only if not using an IAM instance role.** The AWS IAM access key ID that can read from and write to the bucket.
Access key secret	**Required only if not using an IAM instance role.** The AWS IAM secret access key that is associated with the access key ID.

   If using the command line, run the appropriate command:

   Not using an IAM instance role:

```
kubectl kots velero configure-aws-s3 access-key --access-key-id <string>
 --bucket <string> --path <string> --region <string> --secret-access-key
 <string>
```

   Using an IAM instance role:

```
kubectl kots velero configure-aws-s3 instance-role --bucket <string> --
path <string> --region <string>
```

   Azure Blob Storage

   If using the PAM UI, provide the following information:

Field	Description
Note: Only connections via service principals are currently supported.	
Bucket	The name of the Azure Blob Storage container where snapshots are stored.
Path	**Optional.** The path within the container where all snapshots are stored.
Subscription ID	**Required only for access via service principal or AAD Pod Identity.** The subscription ID associated with the target container.
Tenant ID	**Required only for access via service principal .** The tenant ID associated with the Azure account of the target container.
Client ID	**Required only for access via service principal .** The client ID of a Service Principle with access to the target container.
Client secret	**Required only for access via service principal .** The Client Secret of a Service Principle with access to the target container.

4. Click **Update storage settings** to save your storage destination information.

   Depending on your chosen storage provider, saving and configuring your storage provider might take several minutes.
5. Optional: To automatically create new snapshots on a schedule, select **Enable automatic scheduled snapshots** on the **Full snapshots (instance)** tab. (If desired, you can also set up a schedule for capturing partial (application-only) snapshots.)

   You can schedule a new snapshot creation for every hour, day, or week, or you can create a custom schedule by entering a cron expression.
6. Finally, set the retention schedule for your snapshots by selecting the time period after which old snapshots are automatically deleted. The default retention period is one month.

   > **Note:** A snapshot's retention period cannot be changed once the snapshot is created. If you update the retention schedule, the new retention period applies only to snapshots created after the update is made.
7. Click **Update schedule** to save your changes.

Snapshots are automatically created according to your specified schedule and saved to the storage location you selected. You can also create an unscheduled snapshot at any time by clicking **Start a snapshot** on the **Dashboard** or on the **Snapshots** page.

Roll back changes using a snapshot

When necessary, you can use a snapshot to roll back to a previous version of your Puppet Application Manager set-up without changing the underlying cluster infrastructure.

To roll back changes:

1. In console menu of the Puppet Application Manager UI, click **Snapshots** > **Full Snapshots (Instance)**.
2. Select the snapshot you wish to roll back to from the list of available snapshots and click **Restore from this backup**  .
3. Follow the instructions to complete either a partial restore or a full restore.

   A full restore is useful if you need to stay on an earlier version of an application and want to disable automatic version updates. Otherwise, a partial restore is the quicker option.

Migrating PAM data to a new system

By using a snapshot, you can migrate your data to a new Puppet Application Manager (PAM) instance.

Data migration prerequisites

In order to perform a data migration, your system must be configured as follows:

- On the original system, Puppet Application Manager (PAM) must be configured to support **Full Snapshots (Instance)**. For instructions on configuring snapshots, see Backing up PAM using snapshots on page 73.
- Velero must be configured to use an external snapshot destination accessible to both the old and new clusters, such as S3 or NFS.
- Both the old and new clusters must have the same connection status (online or offline). Migrating from offline to online clusters or vice versa is not supported.
- For offline installs, both the old and new clusters must use the same version of PAM.
- Upgrade to the latest version of PAM on both the old and new clusters before you begin.

Migrating data between two systems with the same architecture

To perform data migration between two systems using the same architecture (from standalone to standalone, or from HA to HA), you must create a new cluster to migrate to, then follow the process outlined below to recover your instance from a snapshot.

Before you begin

Review the requirements in Data migration prerequisites on page 75.

Important: If you are migrating from a legacy architecture, go to our Support Knowledge Base instructions for migrating to a supported architecture for your Puppet application:

- Migrate to a supported PAM architecture for Continuous Delivery for PE
- Migrate to a supported PAM architecture for Comply

1. On the original system, find the version of kURL your deployment is using by running the following command. Save the version for use in step 3.

```
kubectl get configmap -n kurl kurl-current-config -o
 jsonpath="{.data.kurl-version}" && echo
```

2. Get the installer `spec` section by running the command appropriate to your PAM installation type:

> **Tip:** See How to determine your version of Puppet Application Manager if you're not sure which installation type you're running.

- HA installation: `kubectl get installers puppet-application-manager -o yaml`
- Standalone installation: `kubectl get installers puppet-application-manager-standalone -o yaml`
- Legacy installation: `kubectl get installers puppet-application-manager-legacy -o yaml`

The command's output looks similar to the following. The `spec` section is shown in bold in the example below. Save your `spec` section for use in step 3.

```
# kubectl get installers puppet-application-manager-standalone -o yaml
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

  {"apiVersion":"cluster.kurl.sh/v1beta1","kind":"Installer","metadata":
{"annotations":{},"creationTimestamp":null,"name":"puppet-application-
manager-standalone","namespace":"default"},"spec":{"containerd":
{"version":"1.4.12"},"contour":{"version":"1.18.0"},"ekco":
{"version":"0.16.0"},"kotsadm":{"applicationSlug":"puppet-
application-manager","version":"1.64.0"},"kubernetes":
{"version":"1.21.8"},"metricsServer":{"version":"0.4.1"},"minio":
{"version":"2020-01-25T02-50-51Z"},"openebs":
{"isLocalPVEnabled":true,"localPVStorageClassName":"default","version":"2.6.0"},"prom
{"version":"0.49.0-17.1.1"},"registry":{"version":"2.7.1"},"velero":
{"version":"1.6.2"},"weave":
{"podCidrRange":"/22","version":"2.8.1"}},"status":{}}
  creationTimestamp: "2021-06-04T00:05:08Z"
  generation: 4
  labels:
    velero.io/exclude-from-backup: "true"
  name: puppet-application-manager-standalone
  namespace: default
  resourceVersion: "102061068"
  uid: 4e7f1196-5fab-4072-9399-15d18dcc5137
spec:
  containerd:
    version: 1.4.12
  contour:
    version: 1.18.0
  ekco:
    version: 0.16.0
  kotsadm:
    applicationSlug: puppet-application-manager
    version: 1.64.0
  kubernetes:
    version: 1.21.8
  metricsServer:
    version: 0.4.1
  minio:
    version: 2020-01-25T02-50-51Z
  openebs:
    isLocalPVEnabled: true
    localPVStorageClassName: default
    version: 2.6.0
  prometheus:
    version: 0.49.0-17.1.1
  registry:
    version: 2.7.1
  velero:
    version: 1.6.2
  weave:
```

3. On a new machine, create a file named `installer.yaml` with the following contents, replacing `<SPEC>` and `<KURL VERSION>` with the information you gathered in the previous steps.

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
<SPEC>
  kurl:
    installerVersion: "<KURL VERSION>"
```

> **Important:** If you are running PAM version 1.68.0 or newer, the kURL installer version might be included in the `spec` section. If this is the case, omit the `kurl:` section from the bottom of the `installer.yaml` file. There must be only one `kurl:` entry in the file.

> **Tip:** Spacing is critical in YAML files. Use a YAML file linter to confirm that the format of your file is correct.

Here is an example of the contents of the `installer.yaml` file:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
spec:
  containerd:
    version: 1.4.12
  contour:
    version: 1.18.0
  ekco:
    version: 0.16.0
  kotsadm:
    applicationSlug: puppet-application-manager
    version: 1.64.0
  kubernetes:
    version: 1.21.8
  metricsServer:
    version: 0.4.1
  minio:
    version: 2020-01-25T02-50-51Z
  openebs:
    isLocalPVEnabled: true
    localPVStorageClassName: default
    version: 2.6.0
  prometheus:
    version: 0.49.0-17.1.1
  registry:
    version: 2.7.1
  velero:
    version: 1.6.2
  weave:
    podCidrRange: /22
    version: 2.8.1
  kurl:
    installerVersion: "v2022.03.11-0"
```

4. Build an installer using the `installer.yaml` file. Run the following command:

```
curl -s -X POST -H "Content-Type: text/yaml" --data-binary
 "@installer.yaml" https://kurl.sh/installer |grep -o "[^/]*$"
```

The output is a hash. Carefully save the hash for use in step 5.

5. Install a new cluster. To do so, you can either:
   a. Point your browser to `https://kurl.sh/<HASH>` (replacing `<HASH>` with the hash you generated in step 4) to see customized installation scripts and information.
   b. Follow the appropriate PAM documentation.
      - **For online installations:** Follow the steps in PAM HA online installation on page 44 or PAM standalone online installation on page 51, replacing the installation script with the following:

        ```
        curl https://kurl.sh/<HASH> | sudo bash
        ```

      - **For offline installations:** Follow the steps in PAM HA offline installation on page 48 or PAM standalone offline installation on page 54, replacing the installation script with the following:

        ```
        curl -LO https://k8s.kurl.sh/bundle/<HASH>.tar.gz
        ```

        When setting up a new *offline* cluster as part of disaster recovery, add `kurl-registry-ip=<IP>` to the install options, replacing `<IP>` with the value you recorded when setting up snapshots.

        > **Note:** If you do not include the `kurl-registry-ip=<IP>` flag, the registry service will be assigned a new IP address that does not match the IP on the machine where the snapshot was created. You must align the registry service IP address on the new offline cluster to ensure that the restored configuration can pull images from the correct location.

> **Important:** Do not install any Puppet applications after the PAM installation is complete. You'll recover your Puppet applications later in the process.

6. To recover using a snapshot saved to a **host path**, ensure user/group 1001 has full access on all nodes by running:

   ```
   chown -R 1001:1001 /<PATH/TO/HOSTPATH>
   ```

7. Configure the new cluster to connect to your snapshot storage location. Run the following to see the arguments needed to complete this task:

   ```
   kubectl kots -n default velero configure-{hostpath,nfs,aws-s3,other-
   s3,gcp} --help
   ```

8. Run `kubectl kots get backup` and wait for the list of snapshots to become available. This might take several minutes.
9. Start the restoration process by running `kubectl kots restore --from-backup <BACKUP NAME>`.

   The restoration process takes several minutes to complete. When the PAM UI is available, use it to monitor the status of the application.

> **Note:** When restoring, wait for all restores to complete before making any changes. The following command waits for pods to finish restoring data from backup. Other pods may not be ready until updated configuration is deployed in the next step:
>
> ```
> kubectl get pod -o json | jq -r '.items[] |
>  select(.metadata.annotations."backup.velero.io/backup-volumes")
>  | .metadata.name' | xargs kubectl wait --for=condition=Ready pod --
> timeout=20m
> ```
>
> This command requires the jq CLI tool to be installed. It is available in most Linux OS repositories.

10. After the restoration process completes, save your config and deploy:

a) From the PAM UI, click **Config**.
b) (Optional) If the new cluster's hostname is different from the old one, update the **Hostname**.
c) Click **Save Config**.
d) Deploy the application. You must save your config and deploy even if you haven't made any changes.

> **Note:** If you have installed Continuous Delivery for PE and changed the hostname, you need to update the webhooks that connect Continuous Delivery for PE with your source control provider. For information on how to do this, see Update webhooks.

Migrating data between two systems with different architectures

To perform data migration between two systems using different PAM architectures (from standalone to HA, or from HA to standalone), you must create a new cluster to recover to, then follow the process outlined below to recover your instance from a snapshot.

> **Before you begin**
>
> Review the requirements in Data migration prerequisites on page 75.
>
> **Important: If you are migrating from a legacy architecture**, go to our Support Knowledge Base instructions for migrating to a supported architecture for your Puppet application:
>
> - Migrate to a supported PAM architecture for Continuous Delivery for PE
> - Migrate to a supported PAM architecture for Comply

1. On the original system, find the version of kURL your deployment is using by running the following command. Save the version for use in step 2.

```
kubectl get configmap -n kurl kurl-current-config -o
 jsonpath="{.data.kurl-version}" && echo
```

2. Set up a new cluster to house the recovered instance, following the system requirements for your applications.

> **Important:** Do not install any Puppet applications after the PAM installation is complete. You'll recover your Puppet applications later in the process.

- Install PAM using the version of kURL you retrieved earlier:
  - For online installs:

    ```
    curl -sSL https://k8s.kurl.sh/version/<VERSION STRING>/puppet-
    application-manager | sudo bash <-s options>
    ```

  - For offline installs:

    ```
    curl -O https://k8s.kurl.sh/bundle/version/<VERSION STRING>/puppet-
    application-manager.tar.gz
    ```

- When setting up a new *offline* cluster as part of disaster recovery, add `kurl-registry-ip=<IP>` to the install options, replacing `<IP>` with the value you recorded when setting up snapshots.

> **Note:** If you do not include the `kurl-registry-ip=<IP>` flag, the registry service will be assigned a new IP address that does not match the IP on the machine where the snapshot was created. You must align the registry service IP address on the new offline cluster to ensure that the restored configuration can pull images from the correct location.

3. To recover using a snapshot saved to a **host path**, ensure user/group 1001 has full access on all nodes by running:

```
chown -R 1001:1001 /<PATH/TO/HOSTPATH>
```

4. Configure the new cluster to connect to your snapshot storage location. Run the following to see the arguments needed to complete this task:

```
kubectl kots -n default velero configure-{hostpath,nfs,aws-s3,other-
s3,gcp} --help
```

5. Run `kubectl kots get backup` and wait for the list of snapshots to become available. This might take several minutes.
6. Start the restoration process by running `kubectl kots restore --from-backup <BACKUP NAME>`.

   The restoration process takes several minutes to complete. When the PAM UI is available, use it to monitor the status of the application.

> **Note:** When restoring, wait for all restores to complete before making any changes. The following command waits for pods to finish restoring data from backup. Other pods may not be ready until updated configuration is deployed in the next step:
>
> ```
> kubectl get pod -o json | jq -r '.items[] |
>  select(.metadata.annotations."backup.velero.io/backup-volumes")
>  | .metadata.name' | xargs kubectl wait --for=condition=Ready pod --
> timeout=20m
> ```
>
> This command requires the jq CLI tool to be installed. It is available in most Linux OS repositories.

7. After the restoration process completes, save your config and deploy:
   a) From the PAM UI, click **Config**.
   b) (Optional) If the new cluster's hostname is different from the old one, update the **Hostname**.
   c) Click **Save Config**.
   d) Deploy the application. You must save your config and deploy even if you haven't made any changes.

   > **Note:** If you have installed Continuous Delivery for PE and changed the hostname, you need to update the webhooks that connect Continuous Delivery for PE with your source control provider. For information on how to do this, see Update webhooks.

Disaster recovery with PAM

It is important to prepare your system and regularly capture full snapshots. This backs up your data and makes it easier to restore your system if disaster recovery is needed.

Prepare your system to support future disaster recovery

To make sure your system is equipped to help you recover from a potential system failure, you must:

- Configure Puppet Application Manager (PAM) to support **Full Snapshots (Instance)**. For instructions on configuring snapshots, see Backing up PAM using snapshots on page 73.
- Configure Velero to use an external snapshot destination that is accessible to both your current cluster and future new clusters, such as S3 or NFS.

- Disaster recovery requires that the store backend used for backups is accessible from the new cluster. When setting up snapshots in an *offline* cluster, use the following command to record the registry service IP address:

  ```
  kubectl -n kurl get svc registry -o jsonpath='{.spec.clusterIP}'
  ```

  Make a record of the value returned by this command, because you'll need it to create a new cluster to restore to as part of disaster recovery.
- Run the latest version of PAM. Disaster recovery is only available on systems running PAM version 1.44.1 or newer.

Disaster recovery process

To perform a disaster recovery, you must create a new cluster to recover to and then recover your instance from a snapshot.

1. Find the version of kURL your original deployment was using.

   If you have access to the original cluster, you can use this command:

   ```
   kubectl get configmap -n kurl kurl-current-config -o
    jsonpath="{.data.kurl-version}" && echo
   ```

   If you aren't able to run the command, you remember your PAM version, and you were on version 1.68.0 or later, you can look up the kURL version in the Component versions in PAM releases on page 40 table.

   If you don't remember your PAM version or you were on a version earlier than 1.68.0, you need to contact your technical account manager or Support for assistance.
2. If you have access to the original cluster, follow the steps for Migrating data between two systems with the same architecture on page 75.

   If your original cluster is completely offline and inaccessible, you'll need to contact your technical account manager or Support for assistance.

> **Restriction:**
>
> Your old and new clusters must have the same connection status (online or offline). Disaster recovery from an offline to an online cluster (or vice versa) is not supported.
>
> Additionally, for offline installs, both the old and new clusters must use the same PAM version.

Troubleshooting PAM

Use this guide to troubleshoot issues with your Puppet Application Manager installation.

How to look up your Puppet Application Manager architecture

If you're running PAM on a Puppet-supported cluster, you can use the following command to determine your PAM architecture version:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o
 jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which architecture you used when installing, the command returns one of these values:

- **HA architecture**: `puppet-application-manager`
- **Standalone architecture**: `puppet-application-manager-standalone`
- **Legacy architecture**: Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`

Resolve IP address range conflicts

When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

> **Note:** The minimum size for CIDR blocks used by Puppet Application Manager are:
>
> - **Standalone** - /24 for pod and service CIDRs
> - **HA** - /23 for pod and service CIDRs
> - Default of /22 is recommended to support future expansion

To resolve IP address range conflicts, create a `patch.yaml` file and add the `installer-spec-file=patch.yaml` argument when running the installation script (see below):

1. If you use IP addresses internally that overlap **`10.32.0.0/22`**, add the following to your `patch.yaml` file (`10.40.0.0/23` used here as an example range):

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  flannel:
    podCIDR: 10.40.0.0/23
    podCIDRRange: "/22"
```

2. If you use IP addresses internally that overlap **`10.96.0.0/22`**, add the following to your `patch.yaml` file (`10.100.0.0/23` used here as an example range):

```
spec:
  ...
  kubernetes:
    serviceCIDR: 10.100.0.0/23
    serviceCIDRRange: "/23"
```

> **CAUTION:** The `podCIDR` and `serviceCIDR` ranges must not overlap.

3. Once your `patch.yaml` file is set up, add the `installer-spec-file=patch.yaml` argument when you run the installation script:

```
cat install.sh | sudo bash -s airgap installer-spec-file=patch.yaml
```

> **Remember:** Add the `installer-spec-file=patch.yaml` argument any time you re-run the installation script, such as when reinstalling to upgrade to a new version.

Related information

Using sudo behind a proxy server on page 86
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Reset the PAM password

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

1. To reset the Puppet Application Manager password, run the following command as the root user:

   ```
   kubectl kots reset-password default
   ```

   The system prompts you to enter a new password of your choosing.
2. If the command fails with an `unknown command "kots" for "kubectl"` error, it's because `/usr/local/bin` is not in the path. To address this error, either update the path to include `/usr/local/bin`, or run this command:

   ```
   /usr/local/bin/kubectl-kots reset-password default
   ```

Update the PAM TLS certificate

A self-signed TLS certificate secures the connection between your browser and Puppet Application Manager (PAM). Once the initial Puppet Application Manager setup process is complete, you can upload new certificates by enabling changes to the installation's Kubernetes secrets.

Use this process if you chose not to add a TLS certificate when installing Puppet Application Manager, or if you need to update your existing TLS certificate.

1. Enable changes to your installation's `kotsadm-tls` Kubernetes secret by running:

   ```
   kubectl -n default annotate secret kotsadm-tls acceptAnonymousUploads=1
   ```
2. Restart the `kurl-proxy` pod to deploy the change by running:

   ```
   kubectl delete pods $(kubectl get pods -A | grep kurl-proxy | awk '{print
    $2}')
   ```
3. Once the kurl-rpoxy pod restarts and is back up and running, navigate to `https://<HOSTNAME>:8800/tls` and upload your new TLS certificate.

Reduce recovery time when a node fails

If a node running a non-replicated service like PostgreSQL fails, expect some service downtime.

How much downtime depends on the following factors:

- Timeout for communication between Kubernetes services (at least one minute to mark the node as unreachable).
- Timeout for the ekco service to determine that pods need to be rescheduled. The default is five minutes after node is marked unreachable.
- Time to restart services (at least two minutes, possibly up to five minutes, if there are complex dependencies).

The ekco service can be configured to reschedule pods more quickly by configuring the installation with a `patch.yaml`similar to the following:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  ekco:
    nodeUnreachableToleration: 1m
```

Apply the patch during an install or upgrade by including `installer-spec-file=patch.yaml` as an install option.

> **Important:** This patch needs to be included during all future upgrades to avoid resetting the option.

PAM components

Puppet Application Manager (PAM) uses a range of mandatory and optional components.

Support services

- Database: PostgreSQL (single instance) - https://www.postgresql.org/
- Object storage: previously MinIO - https://min.io, now Ceph - https://ceph.io
- tlser for basic TLS cert management - https://github.com/puppetlabs/tlser
- kurl_proxy for HTTPS proxying outside the Ingress (ports besides 80/443): https://github.com/replicatedhq/kots/tree/v1.36.1/kurl_proxy

Kubernetes components

- Networking (CNI): Flannel - https://github.com/flannel-io/flannel
- Storage (CSI): Rook - https://rook.io, Ceph - https://ceph.io
- Ingress: Project Contour - https://projectcontour.io
- Kubernetes Cluster: kURL - https://kurl.sh
- Embedded kURL Cluster Operator: ekco - https://github.com/replicatedhq/ekco
- Admin Console: KOTS - https://kots.io
- Snapshots: Velero - https://velero.io, Restic - https://restic.net
- Monitoring: Prometheus - https://prometheus.io
- Registry: Docker Registry - https://docs.docker.com/registry/

Optional components

Prometheus (+Grafana) and Velero (+Restic) are optional components:

- Prometheus+Grafana uses 112m/node + 600m CPU, 200MiB/node + 1750MiB RAM
- Velero+Restic uses 500m/node + 500m CPU, 512MiB/node + 128MiB RAM

If you do not need these optional components, they can be omitted from the initial install and further upgrades with a patch similar to the following:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  prometheus:
    version: ''
  velero:
    version: ''
```

Important: This patch needs to be included during upgrades to avoid adding the components later.

If you want to remove optional components that are already installed, use the following command:

```
kubectl delete ns/monitoring ns/velero
```

Generate a support bundle

When seeking support, you might be asked to generate and provide a support bundle. This bundle collects a large amount of logs, system information and application diagnostics.

To create a support bundle:

1. In Puppet Application Manager UI, click **Troubleshoot** > **Generate a support bundle**.

2. Select a method for generating the support bundle:
   - **Generate the bundle automatically.** Click **Analyze <APPLICATION NAME>** (<APPLICATION NAME> is replaced in the UI by the name of the Puppet application you have installed), and Puppet Application Manager generates the bundle for you and uploads it to the **Troubleshoot** page.
   - **Generate the bundle manually.** Click the prompt to generate a custom command for your installation, then run the command on your cluster. Follow the prompts to upload the bundle to Puppet Application Manager.
3. Review the collected data before forwarding it to Puppet, as it may contain sensitive information that you wish to redact.
4. Return to the **Troubleshoot** page, download the newly created support bundle, and send it to your Puppet Support contact.

Create a support bundle from the command line

If installation of the Puppet Application Manager, or upload of an app, on an embedded kURL cluster fails, it may not be possible to access the UI to generate a support bundle.

You can generate a support bundle by using the default kots.io spec. To do this, run the following command:

```
kubectl support-bundle https://kots.io
```

On an offline server, you can copy the default kots.io spec by using the following command:

```
curl -o spec.yaml https://kots.io -H 'User-agent:Replicated_Troubleshoot/
v1beta1'
```

The spec can then be uploaded to the server. Use the local spec by running:

```
kubectl support-bundle /path/to/spec.yaml
```

If the Puppet Application Manager UI is working and the app is installed, you can use:

```
kubectl support-bundle http://<server-address>:8800/api/v1/troubleshoot/
<app-slug>
```

If the app is not installed but the Puppet Application Manager UI is running:

```
kubectl support-bundle http://<server-address>:8800/api/v1/troubleshoot
```

If you do not already have the support-bundle kubectl plugin installed, install it by using the command below:

```
curl https://krew.sh/support-bundle | bash
```

Or by installing krew2 and running:

```
kubectl krew install support-bundle
```

Using sudo behind a proxy server

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

kURL can only be upgraded two minor versions at a time

Because kURL does not support upgrading more than two Kubernetes minor release versions at once, if you're upgrading from an older version of PAM, you might need to follow a specific upgrade path to avoid failures. For example, PAM version 1.80.0 uses Kubernetes version 1.**21**.x, so you can upgrade up to PAM 1.91.3 (Kubernetes

version 1.**23**.x), but not to PAM 1.94.0 (Kubernetes version 1.**24**.x). To determine the specific upgrade path for your installation, please check the table of Kubernetes versions for each version of PAM.

Attempting to upgrade too far at once returns the following error message: `The currently installed kubernetes version is <CURRENT VERSION>. The requested version to upgrade to is <INVALID_TARGET_VERSION>. Kurl can only be upgraded two minor versions at time. Please install <VALID_TARGET_VERSION> first.`

Set up Comply

To start using Puppet Comply, you must complete the setup process, using both Comply and Puppet Enterprise (PE).

> **Important:** Before you set up Comply, make sure you have installed Puppet Application Manager and reviewed the Comply and PE system requirements.

Setting up Comply involves the following steps:

- Deploy and configure Comply
- Install the `comply` module
- Download the CIS assessor file
- Classify the nodes you want to scan in Puppet Enterprise (PE)
- Configure Comply to communicate with PE

Comply and PE system requirements

Refer to these system requirements to allow your Puppet Comply application to connect to Puppet Enterprise (PE).

Open port requirements

Comply is deployed on a Kubernetes cluster, which requires the following ports:

Port	Protocol	Purpose	Source	Destination
PE ports				
8143	TCP	PE integration	Comply	PE Orchestrator
8081	TCP	PE integration	Comply	PuppetDB
4433	TCP	PE integration	Comply	PE RBAC
Comply ports				
443	TCP	Comply access	User browser	Comply UI
443	TCP	Sending reports	Scan target node	Comply serve

Supported Puppet Enterprise versions

The following versions of Puppet Enterprise (PE) are supported for use with Comply:

PE version
2019.8.4 and later

For more on PE versions, see Puppet Enterprise support lifecycle.

Configure and deploy Comply in an online environment

The Comply installation process creates a Kubernetes cluster and installs the application on that cluster.

Before you begin

Follow the instructions to install Puppet Application Manager.

1. In Puppet Application Manager, upload your Comply license and follow the prompts.

   You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets Comply system requirements.

   Note: The license file is issued by Puppet. If you do not have a license file, contact your Puppet representative. You must also agree to our license agreement. If your licence terms update, for example the expiry date or number of licensed nodes, upload your updated license file to Puppet Application Manager.

2. To configure your installation, click **Config**.
   a) In the **Hostname** field, enter the fully qualified domain name (FQDN) that you want to use to access Comply.

      For example, this could be the name of the node you have installed Comply on. If you choose to use an FQDN that is different from the name of this node, you must configure your domain name system (DNS) to resolve the FQDN to the IP address of the Comply node.
   b) Configure any other settings on the page relevant to your installation.
   c) When you have finished making any necessary changes to the configuration, click **Continue**.
3. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while your system is checked to make sure your cluster meets minimum system requirements. When the preflight check is complete:
   - If the status is **Checks Failed**, click **View preflights**. Correct the issues and click **Re-run**. Repeat this step as needed.

     Important: Do not move on until all preflight checks pass.
   - If the status is **Ready to Deploy**, move on to the next step.

4. Once the version is ready to deploy, click **Deploy**. On the **Application** tab, monitor the application for readiness.

   The application's status is shown as **Unavailable** for several minutes while deployment is underway.

   > **Tip:** You can monitor the deployment's progress by running `kubectl get pods --watch`.

   When the deployment is complete, the application status changes to **Ready**.
5. Installation is now complete! Navigate to `https://<COMPLY-HOSTNAME>` using the name of the FQDN you created in step 4a and sign into Comply using the following default credentials:

   Username: comply | **Password**: compliance

   Once you have signed in, set a new password and make a careful note of these credentials.

Install the comply module.

Configure and deploy Comply in an offline environment

Install Puppet Comply in an air-gapped or offline environment where the Comply host server does not have direct access to the internet.

> **Before you begin**
> Follow the instructions to install Puppet Application Manager.

1. In Puppet Application Manager, upload your Comply licence and follow the prompts.

   You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets Comply system requirements.

   > **Note:** The license file is issued by Puppet. If you do not have a license file, contact your Puppet representative. You must also agree to our license agreement. If your licence terms update, for example the expiry date or number of licensed nodes, upload your updated license file to Puppet Application Manager.
2. When prompted, upload the `.airgap` bundle for the most recent version of Comply.

   > **Note:** Puppet provides you with a custom URL from which to download the Comply `.airgap` file. Please open a ticket with Puppet Support to receive your custom download URL.
3. To configure your installation, click **Config**.
   a) In the **Hostname** field, enter the fully qualified domain name (FQDN) that you want to use to access Comply.

      For example, this could be the name of the node you have installed Comply on. If you choose to use an FQDN that is different from the name of this node, you must configure your domain name system (DNS) to resolve the FQDN to the IP address of the Comply node.
   b) Configure any other settings on the page relevant to your installation.
   c) When you have finished making any necessary changes to the configuration, click **Continue**.
4. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while Comply checks your system to make sure your cluster meets minimum system requirements. When the preflight check is complete:

   - If the status is **Checks Failed**, click **View preflights**. Correct the issues and click **Re-run**. Repeat this step as needed.

     > **Important:** Do not move on until all preflight checks pass.
   - If the status is **Ready to Deploy**, move on to the next step.

5. Once the version is ready to deploy, click **Deploy**. On the **Application** tab, monitor the application for readiness.

   The application's status is shown as **Missing** for several minutes while deployment is underway.

   > **Tip:** You can monitor the deployment's progress by running `kubectl get pods --watch`.

   When the deployment is complete, the application status changes to **Ready**.
6. Installation is now complete! Navigate to `https://<COMPLY-HOSTNAME>` using the name of the FQDN you created in step 9a and sign into Comply.

Install the comply module.

Install the `comply` module

Install the `comply` module from Puppet Forge.

> **Before you begin**
> Make sure you've installed Comply.

Modules are self-contained, shareable bundles of code and data. The `comply` module contains a Bolt task — the tool that runs the CIS assessor on your nodes.

The `comply` module lives on Puppet Forge, a repository of thousands of modules. If you're new to PE and Comply, see Managing environment content with a Puppetfile for more information on the Puppetfile and installing modules.

1. Go to the comply module on the Forge.

   Follow the instructions in the **r10k or Code Manager** drop down to add the module declaration to your Puppetfile. You also need to add its dependencies. For example:

   ```
   # Puppet comply module
   mod 'puppetlabs-comply', '1.0.5'

   # dependencies for comply
   mod 'puppet/archive', '4.6.0'
   mod 'puppetlabs/chocolatey', '5.2.1'
   mod 'puppetlabs/inifile', '4.4.0'
   mod 'puppetlabs/java', '6.5.0'
   mod 'puppetlabs/ruby_task_helper', '0.6.0'
   mod 'puppetlabs/stdlib', '6.6.0'
   mod 'puppetlabs/powershell', '4.1.0'
   mod 'puppetlabs/registry', '3.2.0'
   mod 'puppetlabs/pwshlib', '0.8.0'
   ```

   If you don’t specify options, Code Manager installs the latest version and does not update it automatically. To always have the latest version installed, specify `:latest` and it updates automatically when a new version is released. To install a specific version of the module that does not update automatically, specify the version number as a string.
2. SSH into your PE primary server and deploy code by running the `puppet-code deploy --all` command.

Download the CIS assessor file.

Download the CIS assessor file

Download the CIS assessor file from Comply.

> **Before you begin**
> Make sure you've installed Comply and the `comply` module.

> **Note:** The assessor requires Java 1.8 or higher. If Java is not present or has the wrong version on the target node, PE installs or upgrades Java. Java 1.8 requires the node to have at least 2 GB of free RAM.

The `comply` module runs the CIS assessor file on your nodes when scanning. As it is not included in the `comply` module, you must download it separately and before you classify nodes.

1. Navigate to Comply — located at `https://[COMPLY-HOSTNAME]/` — and click **Settings**.
2. Download the CIS assessor file and save it in a location accessible to your nodes.

Classify the nodes you want to scan.

Classify the nodes you want to scan

In Puppet Enterprise (PE), classify the nodes you want to scan.

> **Before you begin**
> Make sure you've installed Comply, the `comply` module, and the CIS assessor file.

Classification is when you create a node group, add nodes to the group, and assign *classes* to the group — in this case, the `comply` class. Classes are the blocks of Puppet code used to configure nodes and assign resources to them. If you are new to Puppet, see Grouping and classifying nodes for more information.

1. In the PE console, click **Node groups**.
2. Create a new node group or select an existing node group that you want to scan.
3. On the **Classes** tab — in the **Add new class** field — select the `comply` class.
4. Click **Add class**.
5. In your new `comply` class, select the `scanner_source` **Parameter**.

   > **Note:** *Parameters* allow a class to request external data.

6. Change the default parameter **Value** to the CIS assessor file that you previously downloaded. For example:

   ```
   https://puppet.company.local:8140/packages/Assessor-CLI-v4.0.24.zip
   ```

   The `scanner_source` parameter now points to the CIS assessor file.

   > **Note:** If you have not downloaded the CIS assessor file, see Download the assessor file for instructions.

7. Click **Add to node group**, and then commit the changes.
8. Run Puppet **twice**.

Add PE credentials.

Add PE credentials to Comply

Puppet Comply is an add-on to Puppet Enterprise (PE). To allow Comply to communicate with PE, you must add your PE credentials to Comply.

> **Before you begin**
> Make sure you've installed Comply and `comply` module, and classified the nodes you want to scan in PE.
>
> Adding your PE credentials authenticates Comply with Role Based Access Control (RBAC). Your PE account requires the following permissions:

Type	Action	Instance
Console	View	-
Job Orchestrator	Start, stop and view jobs	-
Node Groups	View	All
Nodes	View node data from PuppetDB	-
Tasks	Run Tasks	Task: `comply::backup_assessor` Permitted on : All nodes
Tasks	Run Tasks	Task: `comply::ciscat_scan` Permitted on: All nodes

For more information on permissions, see RBAC permissions.

1. Navigate to Comply — located at `https://[COMPLY-HOSTNAME]/` — and click **Settings**.
2. Enter your PE **hostname**, **username**, and **password**.
3. Click **Submit**.

> **Tip:** You can refresh the PE node and fact information by clicking **Refresh data**.

You'll now see a list of your classified nodes on the **Nodes** page.

You have completed the installation and configuration process! You can now start running CIS scans on your nodes. If you're new to Comply, try out the getting started guide.

Upgrading

New versions of Puppet Comply are released regularly. Upgrading to the current version ensures you are always taking advantage of the latest features, fixes, and improvements.

Upgrade Comply in an online environment

Check for download and deploy updates from the **Version history** tab in the Puppet Application Manager UI.

1. In the platform admin console, click **Version history**.
2. Click **Check for updates**.

   Configure an automatic update check by clicking **Configure automatic updates**. You can check for updates hourly, every four hours, daily, weekly, or at a custom interval.
3. If an update is available, Puppet Application Manager downloads it for you and performs preflight checks on your system to make sure your cluster meets system requirements for the new version. Review the outcome of these checks by clicking **View preflight**.
4. When you're ready to upgrade to the new version of Comply, click **Deploy**.

Upgrade Comply in an offline environment

If your environments do not have direct access to the internet, use the links below to upgrade to the latest version of Comply.

1. Navigate to the portal provided to you by Puppet in the licence email, for example, `https://get.replicated.com/airgap/#/kots/comply/`, and login with the password.
2. Select **Embeded cluster** and download the latest Comply release `.airgap` file.

3. Log into Puppet Application Manager — `https://<PLATFORM-ADMIN-CONSOLE-ADDRESS>:8800`.
4. Select **Version history**, and upload the new version of the `.airgap` file that you downloaded in step 2.
5. Click **Deploy**.

Upgrade the `comply` module

Upgrade to the latest version of the `comply` module in Puppet Enterprise (PE).

> **Note:** Take note of module dependencies when upgrading to a new major version — you need to upgrade these as well.

1. Update your Puppetfile with the latest version of the `comply` module and its dependencies. For example:

```
# Puppet comply module
mod 'puppetlabs-comply', '1.0.5'

# dependencies for comply
mod 'puppet/archive', '4.6.0'
mod 'puppetlabs/chocolatey', '5.2.1'
mod 'puppetlabs/inifile', '4.4.0'
mod 'puppetlabs/java', '6.5.0'
mod 'puppetlabs/ruby_task_helper', '0.6.0'
mod 'puppetlabs/stdlib', '6.6.0'
mod 'puppetlabs/powershell', '4.1.0'
mod 'puppetlabs/registry', '3.2.0'
mod 'puppetlabs/pwshlib', '0.8.0'
```

2. SSH into your PE primary server and deploy code by running the `puppet-code deploy --all` command.

Upgrade the CIS assessor

When you upgrade the `comply` module, new nodes are classified with the latest CIS assessor version. Existing nodes stay on the previous CIS assessor version. This lets you select when you want to upgrade the assessor on each node.

1. Navigate to Comply — located at `https://[COMPLY-HOSTNAME]/` — and click **Settings**.
2. Download the new CIS assessor file and save it in a location accessible to your nodes.
3. Navigate to Puppet Enterprise (PE), and click **Node groups**.
4. Update the `scanner_source` parameter of your `comply` class node group with the new assessor file.
5. Run Puppet to install the latest version of the assessor.

> **Note:** If this fails, you can revert changes to the `scanner_source` parameter of your `comply` class node group to get the old assessor back.

6. When you have upgraded the new assessor, remove the backup assessor. Run `comply::backup_assessor` task with the `operation = delete` parameter.

Reset desired compliance to use the latest CIS benchmarks.

Uninstall Comply

Uninstall Comply by deleting the Comply application and purging the Kubernetes cluster.

1. From the command line of the node where you have Comply installed, run the following command to delete the Comply application:

```
# kubectl delete $(kubectl api-resources --verbs=delete -o name | paste -
sd "," -) -A -l app.kubernetes.io/part-of=comply
```

2. On the same node, run the following command to uninstall the embedded Kubernetes cluster:

```
curl https://k8s.kurl.sh/comply/tasks.sh | sudo bash -s reset
```

> **Note:** This command resets the Replicated installation with a purge.

3. Reboot your node to clear the `kube-ipvs0` device.

Desired compliance

Set your desired compliance. This is the benchmark and profile that you to assign to a particular node. It is what is scanned on that node by default. Most of the time, you only need to set this once for your nodes.

There are two ways to specify desired compliance:

- **Option 1: Allow Comply to automatically set desired compliance.** Based on fact information from PE, Comply can automatically assign an appropriate benchmark for each operating system, along with a Level 1 profile, to nodes that have not been set. This is the quickest way to get up and running with desired compliance.
- **Option 2: Manually set desired compliance.** If you want to choose a specific benchmark and profile for your nodes, or even a custom profile, option two provides this flexibility.

> **Note:** Only one benchmark and profile can be assigned to each node.

Option 1: Allow Comply to automatically set desired compliance

1. In Comply, click **Nodes**.

   Comply lists the nodes that have been classified with the `comply` class. If you do not see any nodes, ensure you have classified your nodes correctly.
2. In the message box that appears in the top right corner, click **Apply suggested profiles**.

   Comply automatically assigns the appropriate benchmark, along with a Level 1 profile, to all the nodes that have not already been set, on your *current* page. To apply the suggested profile to all the nodes in your inventory, you must do this on every page.

   > **Tip:** If you want to customize your scans to fit your organization's internally defined standards, see Creating custom profiles, which shows you how to exclude rules in a profile.

The ## sign in the **Profile assigned** column tells you that the desired compliance is set. You can view the node's information, including its assigned benchmark and profile, by clicking on the node. If you want to change a node's desired compliance, use the drop-down menu and click **Update**.

Now that you have applied desired compliance, you can use this option to run a CIS scans.

Option 2: Manually set desired compliance

1. In Comply, click **Nodes**.

   Comply lists the nodes that have been classified with the `comply` class. If you do not see any nodes, ensure you have classified your nodes correctly.
2. Click on the node for which you want to specify desired compliance.

   In the window that appears on the right, you can see facts about the node and whether desired compliance has been set.

3. Choose the CIS benchmark and profile that you want to assign to the node.

   The benchmark and profile you set here is the desired compliance option for future scans.

   If you have created a custom profile, you can set it as the desired compliance by clicking **Use an associated custom profile?**.
4. Click **Update**.

The ## sign in the **Profile assigned** column tells you that the desired compliance is set. You can view the node's information, including its assigned benchmark and profile, by clicking on the node. If you want to change a node's desired compliance, use the drop-down menu and click **Update**.

Now that you have applied desired compliance, you can use this option to run scans.

Related information

Create a custom profile on page 95
Create a custom profile based on an existing benchmark.

Run a CIS scan on page 96
Run your desired compliance scan or an ad-hoc scan on your nodes.

Custom profiles

A custom profile is a benchmark profile that you customize to fit your organization's internally defined standards.

You can base a custom profile on an existing benchmark and profile combination, and then specify which rules you want visible in scan reports.

Create a custom profile

Create a custom profile based on an existing benchmark.

1. Navigate to **Custom profiles**.
2. Click **Create custom profile**.
3. Select a **Benchmark** and **Profile**.
4. Deselect rules in the profile that you **do not** want to scan, and click **Next**.
5. Enter the name of the profile and, optionally, a description.
6. Click **Save custom profile**.

   Your custom profile appears as an option when you assign the associated benchmark to a node.

Navigate to **Nodes** to set your custom profile as the desired compliance for your nodes, or preform an ad-hoc scan by selecting your custom profile on the **Scan** page.

Related information

Option 2: Manually set desired compliance on page 94

Run a CIS scan on page 96
Run your desired compliance scan or an ad-hoc scan on your nodes.

CIS scans

Run CIS scans to receive data about the compliance of your nodes.

Run a CIS scan

Run your desired compliance scan or an ad-hoc scan on your nodes.

1. In Comply, click **Scan**.
2. In the **Benchmark** drop-down, select **Desired compliance** or a benchmark and profile of your choice.

   If you have not set desired compliance, see Setting desired compliance for instructions.
3. Click **Next** to review the PE credentials and environment you want the scan to run on.
4. Click **Next** to see the nodes selected for scanning.

   To only scan a subset of nodes, deselect any that you do not want to include.
5. Click **Scan** and then **Start**.

   You'll be taken to the **Activity Feed**, which lists each scan. Scans are run as a task in PE. To see the details of the job, click on the job ID to be taken to PE.

   > **Tip:** You can also run a scan by clicking the **Scan nodes** button at the top right corner on several pages. This option uses the nodes listed on the page you are currently viewing.
6. In Comply, navigate to **Compliance dashboard** to see the results of your scans.

   See Viewing scan results for a description of the scan data.

Related information

Scan results on page 96
View the results of your CIS scans and find out whether your nodes are compliant.

Create a custom profile on page 95
Create a custom profile based on an existing benchmark.

Scan results

View the results of your CIS scans and find out whether your nodes are compliant.

Compliance dashboard

The **Compliance dashboard** provides a breakdown of your latest CIS scan.

It has three components:

- The **Node results** donut shows the percentage of rules that passed and failed across all of your nodes. These percentages are shown with the statuses of **Pass**, **Fail**, **Error**, and **Unknown**.
- The **Desired compliance** donut shows the number of nodes that have desired compliance set. Desired compliance is the default benchmark and profile that you have assigned to that node.
- The **Node results table** lists information about the latest scan for each node.

Node compliance

From the **Compliance dashboard**, click on a node name to get to the **Node compliance** page, and see the results of the latest scan on that node. The data includes:

- The **Rules checked** donut shows a status breakdown for the latest scan on that node — the rules that passed, failed, had an error, or an unknown status.
- The **Rule scan results** table lists each rule that was checked and the status of that rule from the latest scan.

Rule results

From **Node compliance**, click on a rule title for **Rule results** page, and see the details of that rule and the status of the nodes it is checked on. The data includes:

- A tabbed section that displays information about each rule:
  - **Description** — information on what is being checked.
  - **Rationale** — the reason why it is important to check that rule.
  - **Fix** — the steps you can take to fix the rule if it is failing on a node.
- The **Node results** donut shows the nodes that this rule is checked on, and which nodes passed or failed the rule.
- The **Node compliance result** table lists each node the rule has been checked against and shows the current status — when it was last checked and when it last passed that rule.

Related information

Comply terminology on page 5
Key terms to be familiar with when using Puppet Comply.

Desired compliance on page 94
Set your desired compliance. This is the benchmark and profile that you to assign to a particular node. It is what is scanned on that node by default. Most of the time, you only need to set this once for your nodes.

Troubleshooting

Use this section to troubleshoot issues with your Puppet Comply installation.

Reset your Comply password

If you forget your password, you can reset it in the user admin console.

1. SSH into your Comply node and run the following commands to retrieve the admin username and password:

```
kubectl exec $(kubectl get pod -l app.kubernetes.io/name=comply-auth -o
 jsonpath="{.items[0].metadata.name}") -- /bin/bash -c 'cat /etc/keycloak/
admin-user'
```

```
kubectl exec $(kubectl get pod -l app.kubernetes.io/name=comply-auth -o
 jsonpath="{.items[0].metadata.name}") -- /bin/bash -c 'cat /etc/keycloak/
admin-password'
```

2. Navigate to `https://<COMPLY-HOSTNAME>/auth/admin` using the FQDN of your Comply node.
3. Login using the credentials from step 1.
4. Navigate to **Users**.
5. Click **View all users** and select the user account you want to update, and click **Edit**.
6. Select the **Credentials** tab and the reset password.

Access logs

If you run into issues with Puppet Comply, you can download the relevant log files. The Comply logs are stored in Puppet Application Manager.

1. Log into Puppet Application Manager — `https://<PUPPET-APPLICATION-MANAGER-ADDRESS>:8800`.

2. Select the **Troubleshoot** tab, and click **Analyse Comply**.
3. Download the bundle of log files.

Resolve Comply domain

If the Puppet Comply gatekeeper is unable to resolve the Comply domain, try the following troubleshooting steps.

When you assign a hostname to Comply, it needs to be resolved by the pods in your Kubernetes cluster. A preflight check verifies the domain you specified in the configuration is resolvable. You must ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.

1. To verify your whether your hostname is resolvable, run the following commands:

   ```
   kubectl exec $(kubectl get pod -l app=kotsadm -o
    jsonpath="{.items[0].metadata.name}") -- /bin/sh -c 'curl --SI
    <hostname>'
   ```

   If the hostname was resolved, the command returns an exit code 0 with no output.

   If the hostname cannot be resolved, the command returns an exit code 6. Proceed to step 2 to add DNS entries.
2. To add DNS entries for CoreDNS, run the following command to open the CoreDNS configuration maps:

   ```
   kubectl -n kube-system edit configmaps coredns
   ```
3. Add a `hosts` entry below `kubernetes`. This is where you configure the DNS entry for Comply. For example:

   ```
   kubernetes cluster.local in-addr.arpa ip6.arpa {
      pods insecure
      fallthrough in-addr.arpa ip6.arpa
      ttl 30
   }
   hosts {
     10.23.24.25 comply.mycompany.net comply // IP_address canonical_hostname
    [aliases...]
     fallthrough
   }
   prometheus :9153
   ```
4. Run the command from step 1 to verify whether the DNS entry was updated:

   ```
   kubectl exec $(kubectl get pod -l app=kotsadm -o
    jsonpath="{.items[0].metadata.name}") -- /bin/sh -c 'curl --SI
    <hostname>'
   ```
5. Re-run the preflight checks.